UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE TECNOLOGÍA
CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE DESARROLLO MODULAR BASADA EN LA FAMILIA DE MICROCONTROLADORES MCS-51 PARA EL DESARROLLO DE MÚLTIPLES APLICACIONES

PROYECTO DE GRADO PRESENTADO PARA OBTENER EL GRADO DE LICENCIATURA

POR: CRISTIAN FERNANDO LIMACHI PALLY

TUTOR: LIC. EDDER TOMÁS JURADO MOYA

LA PAZ – BOLIVIA

2022

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE TECNOLOGÍA
CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

Proyecto de Grado

DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA DE DESARROLLO MODULAR BASADA EN LA FAMILIA DE MICROCONTROLADORES MCS-51 PARA EL DESARROLLO DE MÚLTIPLES APLICACIONES

Presentado por: Cristian Fernando Limachi Pally

Para optar al grado académico de licenciado en Electrónica y Telecomunicaciones

Nota numeral: ……

Nota literal: ……

Ha sido ……

Lic. Julia Torrez Soria
Directora de la carrera de Electrónica y Telecomunicaciones

Tutor: Lic. Edder Tomás Jurado Moya

Tribunal: Ing. José Arturo Marín Thames

Tribunal: Ing. Luis Ramiro Velarde Chávez

Tribunal: Lic. Juan Carlos Gutiérrez

DEDICATORIA

El presente proyecto va dedicado a mis padres, mis hermanos y todas aquellas personas que creyeron en mí, brindándome su apoyo.

AGRADECIMIENTOS

Agradezco a Dios por haberme bendecido con una familia, quienes a lo largo de toda mi vida han apoyado y motivado mi formación académica, creyeron en mí en todo momento y no dudaron de mis habilidades.

Agradecer también a mi tutor por el apoyo y mis tribunales por las observaciones y correcciones para que este proyecto sea posible.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El propósito de este proyecto es el diseño e implementación de una tarjeta de desarrollo modular para aplicaciones didácticas basada en el núcleo INTEL MCS-51.

Lo que se pretende es poder satisfacer las necesidades de los programadores dando una alternativa la cual pueda expandirse sin problemas, pudiendo tener todo tipo de interfaces interconectadas al microcontrolador principal.

Además de poder usar cualquier compilador sin importar el lenguaje, dando así una ventaja sobre otras tarjetas de desarrollo que tienen un software propio, esto quiere decir que se contara con un cargador de archivos compilados hacia el microcontrolador principal que usará una memoria externa de hasta 64 KB para almacenar los programas.

Para que todo esto sea posible es necesario tener en cuenta el desarrollo de un programa "bootloader" que estará cargado dentro de la memoria del microcontrolador principal y este se encargara de direccionar a la memoria externa los nuevos programas a ejecutarse.

Por otro lado, es necesario tener un cargador de archivos el cual se desarrollará teniendo en cuenta el formato de archivos HEX de Intel para poder interpretar las direcciones y datos a ser enviados.

Haciendo notar lo anterior, se mostrará los pasos para el diseño de la tarjeta propuesta, teniendo los objetivos claros y justificados, además de una parte teórica que ayudará a comprender el diseño y también los términos usados dentro del proyecto, también se tendrá en cuenta los costos de materiales a usar y por último se darán las conclusiones respectivas así también como las recomendaciones.

CAPITULO I
MARCO REFERENCIAL

1.1 ANTECEDENTES

En la búsqueda bibliográfica se han podido rescatar investigaciones y procesos similares que ayudarán a que el presente proyecto rescate todo lo positivo de estos.

El sitio web https://arduino.cl/que-es-arduino/, consultado en 2021. Indica lo siguiente. "Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont)".[1]

El sitio web https://es.wikipedia.org/wiki/BASIC_Stamp, consultado en 2021. Menciona lo siguiente. "BASIC Stamp es un microcontrolador que posee un intérprete especializado de BASIC (PBASIC) que se encuentra en su memoria ROM. Este microcontrolador es fabricado por Parallax, Inc. y es popular entre los aficionados a la electrónica desde principios de la década de 1990 por su facilidad de aprendizaje y su fácil uso, así como el lenguaje de programación BASIC que se requiere para controlar este chip".[2]

El sitio web https://www.hwlibre.com/que-es-una-placa-sbc/, consultado en 2021, indica lo siguiente. "Las siglas SBC quieren decir, Single Board Computer o Pc de placa única. Esto quiere decir que, a diferencia de los ordenadores tradicionales, los pc SBC son placas que contienen todos o la mayor parte de los componentes de un ordenador.

La principal característica de los SBC u ordenadores con placas SBC son sus reducidas dimensiones. Mientras un pc mini-ITX está montado sobre una placa con unas medidas

[1] Arduino, (2021). *¿Qué es Arduino?* Recuperado de https://arduino.cl/que-es-arduino/
[2] Wikipedia, la enciclopedia libre, (2021). *Basic Stamp*. Recuperado de https://es.wikipedia.org/wiki/BASIC_Stamp

de 17 x 17 cm. Aproximadamente, los mini pc o los ordenadores SBC están montados sobre placas con medidas inferiores".[3]

Intel 8051 es un microcontrolador (µC) desarrollado por Intel en 1980 para uso en productos embebidos. Es un microcontrolador muy popular.

Los núcleos 8051 se usan en más de 100 microcontroladores de más de 20 fabricantes independientes como Atmel, Dallas Semiconductor, Philips, Winbond, entre otros.

La denominación oficial de Intel para familia de µCs 8051 es MCS 51.[4]

Los microcontroladores 8051 modernos ofrecen muchas mejoras sobre el original. Mejoras comunes incluyen watchdog timers (un temporizador programable que "resetea" el microcontrolador si no se refresca en cierto tiempo), osciladores internos, memoria de programa Flash ROM interna, código de inicialización en ROM, almacenamiento en EEPROM interna, I²C, SPI, USB, generadores PWM, conversores analógicos A/D y D/A, relojes de tiempo real RTC, temporizadores y contadores extra, facilidades de depuración internas, más fuentes de interrupción, modos de bajo consumo, interfaz CAN, etc.

Se observa que los antecedentes son de tarjetas de desarrollo ya implementadas, ya que aún no existen proyectos similares al que se pretende presentar, por lo que con el presente proyecto se pretende complementar los antecedentes mencionados, adaptándolos a una situación donde se vean varias ventajas respecto a los demás.

También se debe notar que el núcleo MCS-51 aun es implementada en la actualidad teniendo varias empresas fabricantes las cuales incluyen varias mejoras y añadidos.

1.2 PLANTEAMIENTO DEL PROBLEMA

En la última década, las tarjetas de desarrollo han ido tomando mayor importancia por los aficionados a la electrónica o informáticos, dado que estas tarjetas facilitan en gran

[3] Hardware Libre, (2021). *¿Qué es una placa SBC?* Recuperado de https://www.hwlibre.com/que-es-una-placa-sbc/

[4] Wikipedia, la enciclopedia libre, (2021). *Intel 8051*. Recuperado de https://es.wikipedia.org/wiki/Intel_8051

medida el desarrollo de una aplicación que pueda brindar una solución específica a la sociedad. Tarjetas de desarrollo como Arduino, predominan el mercado, por tener la lógica de software y hardware libre, pero estas tienen varias desventajas a la hora de realizar aplicaciones con mayor grado de complejidad, ya que no se da la libertad al programador de poder acceder a todos los registros disponibles del microcontrolador principal.

Por otro lado, también se ve que la expansión de interfaces en Arduino es limitado, por lo que el usuario final debe optar por adquirir otra placa principal con mayor cantidad de interfaces, entre ellas puerto series, o paralelos.

Cabe señalar que las tarjetas de desarrollo por lo general tienen un entorno de programación definido, por lo que no es posible usar diferentes lenguajes o compiladores de terceros.

Lo descrito anteriormente genera un problema de investigación el cual conlleva a las siguientes preguntas principales:

¿Cómo diseñar una tarjeta de desarrollo que sea capaz de expandirse y que el programador pueda acceder a todos los registros del microcontrolador?, ¿De qué manera se podría realizar la implementación?, ¿Cómo hacer que la tarjeta tenga un entorno libre el cual pueda interactuar con cualquier compilador?

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Diseñar e implementar una tarjeta de desarrollo modular basada en la familia de microcontroladores mcs-51 para el desarrollo de múltiples aplicaciones

1.3.2 OBJETIVOS ESPECÍFICOS

- Determinar las características técnicas y económicas del sistema a desarrollar.
- Diseñar la tarjeta de desarrollo propuesta.
- Diseñar el software para cargar los programas a la tarjeta.

- Implementar el prototipo final y corregir errores.

1.4 JUSTIFICACIÓN

El presente proyecto propone diseñar e implementar de una tarjeta de desarrollo modular basada en la familia de microcontroladores mcs-51 para el desarrollo de múltiples aplicaciones, visto que los programadores más experimentados se ven limitados al querer usar la tarjeta de desarrollo Arduino, ya que no pueden acceder a los registros del microcontrolador principal de manera directa, y se ven limitados a estar dentro de un bucle infinito al momento de programar sobre el mismo.

El precio de los componentes necesarios es mucho menor a comparación de otras tarjetas de desarrollo, además que esta tarjeta competiría directamente con un microprocesador de ocho bits al poder asignar más memoria o interfaces dentro de sus direcciones de memoria.

En la actualidad este microcontrolador es usado principalmente en el sector industrial, esto debido a la antigüedad que tiene en el mercado y el hecho de no migrar las aplicaciones ya existentes a otras arquitecturas como PIC, AVR, ARM; ya que esto causaría un gasto para las empresas. En consecuencia, empresas como Phillips y Microchip, han ido actualizando esta arquitectura, manteniendo sus instrucciones, pero también añadiendo nuevas, las cuales potencian al microcontrolador.

En cuanto a los periféricos se refiere, estos aún son fabricados por empresas como NEC, ya que estos son bastante usados en tarjetas de PC.

Tomando en cuenta estos parámetros es que se ve de manera viable el diseño e implementación de una tarjeta de desarrollo modular para diferentes aplicaciones basada en el núcleo MCS-51, teniendo en cuenta el siguiente esquema principal.

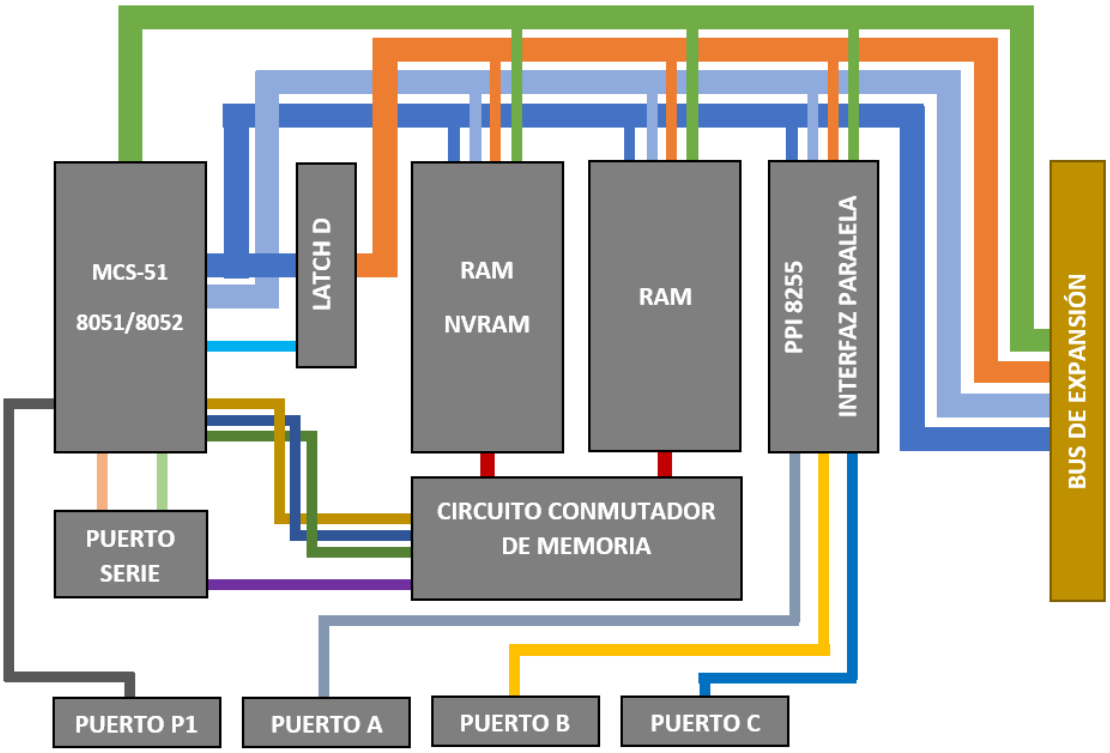


Figura 1: Diagrama de bloques de la placa de desarrollo a ser implementada.

Fuente: Diseño propio.

Se observa que el microcontrolador tiene la propiedad de comportarse como un microprocesador, el cual puede expandirse; combinando ambas características es posible usar el puerto serie del microcontrolador para poder cargar un programa a la memoria externa, implementando así un "bootloader", teniendo cargado el programa principal en la memoria interna del microcontrolador y este a su vez se encargara de cargar nuevos programas a la memoria externa NVRAM, los cuales correrán sin ningún inconveniente y teniendo una disponibilidad de 64KB como máximo.

En cuanto a las aplicaciones se puede mencionar algunos ejemplos de aplicación educativa como encender relés, display de 7 o 14 segmentos, lectura de entradas digitales entre otros. En cuanto a las aplicaciones en un entorno real se puede mencionar control de ascensores, control de motores paso a paso o servos, monitoreo de sensores, comunicaciones de tipo serie, uso de temporizadores; incluso mucho más, dependiendo de los módulos que se puedan implementar en el bus de expansión.

1.5 DELIMITACIONES

1.5.1 DELIMITACIÓN ESPACIAL

El presente proyecto se ve limitado en cuanto a la observación de las limitaciones de programadores electrónicos e informáticos que usan tarjetas de desarrollo convencionales dentro de las instituciones educativas y empresas de la ciudad de La Paz y El Alto.

1.5.2 DELIMITACIÓN TEMPORAL

Por el tiempo de diseño, el proyecto se presentará mostrando solo pruebas base para su entendimiento, sin embargo, la implementación de proyectos complejos sobre esta, será totalmente posible.

1.6 METODOLOGÍA

La metodología del proyecto incluye el tipo o tipos de investigación, las técnicas y los procedimientos que serán utilizados para llevar a cabo la indagación. Es el "como" se realizará el estudio para responder al problema planteado.[5]

La investigación científica es un proceso dirigido a la solución del problema del saber, mediante la obtención de nuevos conocimientos. Dicho proceso comprende las siguientes etapas:[6]

a) Planificación.
b) Ejecución o desarrollo.
c) Divulgación.

El método científico abarca las prácticas aceptadas por la comunidad científica como válidas a la hora de exponer y confirmar sus teorías. Las reglas y principios del método

[5] Fidias G. Arias. (1999). Marco Metodológico. En *El Proyecto de Investigación. Guía para su elaboración* (pág. 45). Caracas: Episteme. Orial Ediciones.
[6] Fidias G. Arias. (1999). Concepto de Investigación. En *El Proyecto de Investigación. Guía para su elaboración* (pág. 22). Caracas: Episteme. Orial Ediciones.

científico buscan minimizar la influencia de la subjetividad del científico en su trabajo, reforzando así la validez de los resultados, y, por ende, del conocimiento obtenido.[7]

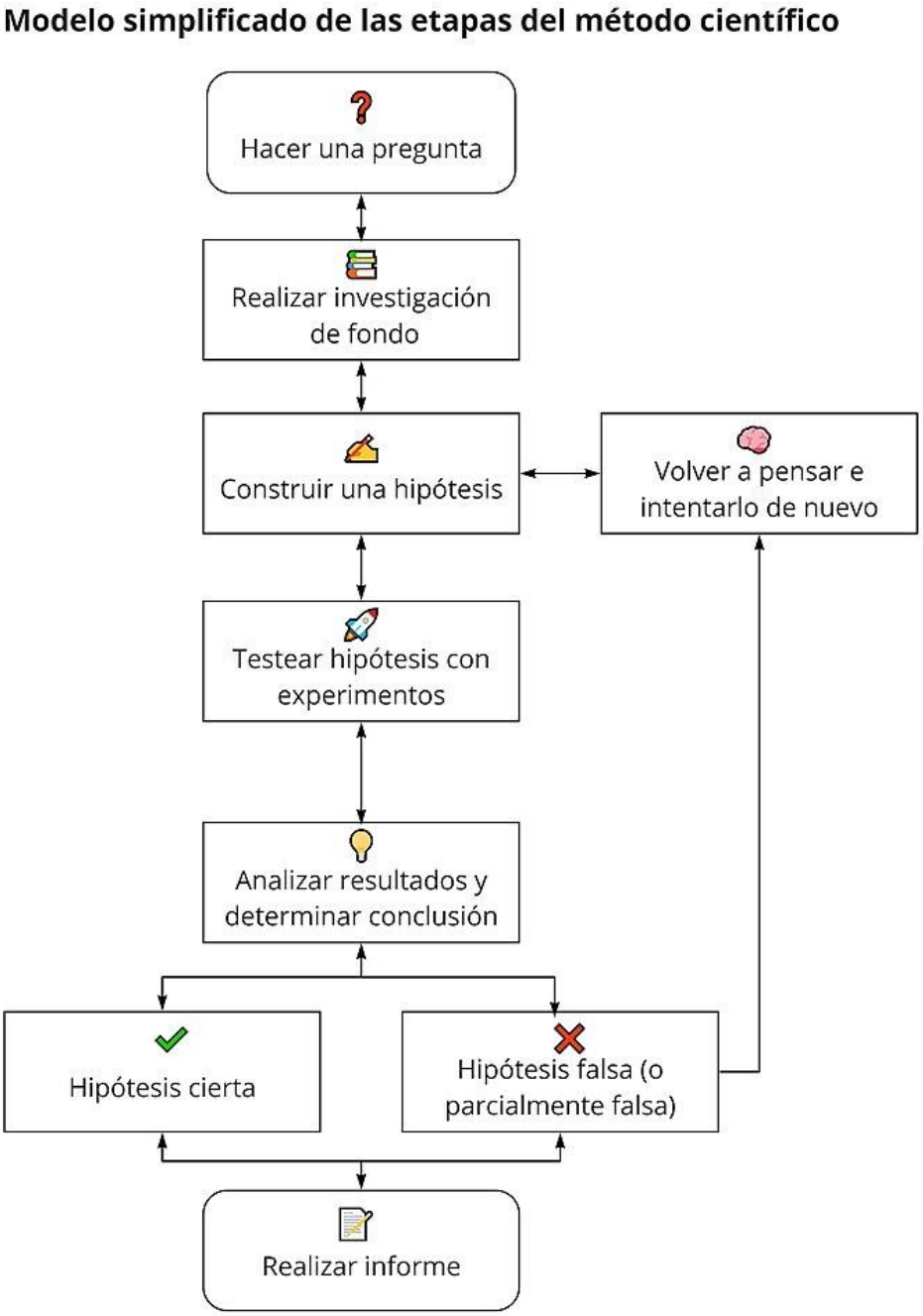


Figura 2: Método simplificado de las etapas del método científico.

Fuente: https://es.wikipedia.org/wiki/M%C3%A9todo_cient%C3%ADfico

[7] Wikipedia, la enciclopedia libre, (2021). *Método Científico*. Recuperado de https://es.wikipedia.org/wiki/M%C3%A9todo_cient%C3%ADfico

CAPITULO II
MARCO TEÓRICO

2.1 SOFTWARE LIBRE

El software libre es un software cuyo código fuente puede ser estudiado, modificado, y utilizado libremente con cualquier finalidad y redistribuido con cambios o mejoras sobre ellas. Su definición está asociada al nacimiento del movimiento de software libre, encabezado por el activista y experto informático estadounidense Richard Stallman y la fundación que presidía en 1985, la Free Software Foundation, una organización sin ánimo de lucro que pone la libertad del usuario informático como propósito ético fundamental.

Un software es libre si otorga a los usuarios de manera adecuada las denominadas cuatro libertades: libertad de usar, estudiar, distribuir y mejorar, de lo contrario no se trata de software libre. Existen diversos esquemas de distribución que no son libres, y si bien podemos distinguirlos sobre la base de cuánto les falta para llegar a ser libres, su uso bien puede ser considerado contrario a la ética en todos los casos por igual.

La expresión «software libre» proviene de la expresión del inglés free software, que presenta ambigüedad entre los significados «libre» y «gratis» asociados a la palabra free. Por esto es que suele ser considerado, de manera errónea, como software gratuito y no en su acepción más precisa como software que puede ser modificado y compartido sin infringir la licencia. El software libre suele estar disponible gratuitamente, o al precio de coste de la distribución a través de otros medios, sin embargo, no es obligatorio que sea así, por lo tanto, no hay que asociar software «libre» a «gratuito» (denominado usualmente freeware), ya que, conservando su carácter de libre, puede ser distribuido comercialmente. Análogamente, el software gratis o gratuito incluye en ocasiones el código fuente; no obstante, este tipo de software no es «libre» en el mismo sentido que el software libre, a menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa. En este sentido, es importante conocer las implicaciones jurídicas que emanan del uso del software libre.

Tampoco debe confundirse software libre con «software de dominio público». Este último es aquel que no requiere de licencia, pues sus derechos de explotación son para toda la humanidad, porque permite el acceso a todos por igual. Cualquiera puede hacer uso de él, consignando su autoría original. Este software sería aquel cuyo autor lo dona a la humanidad o cuyos derechos de autor han expirado. Si un autor condiciona su uso bajo una licencia, por muy débil que sea, ya no es del dominio público.[8]

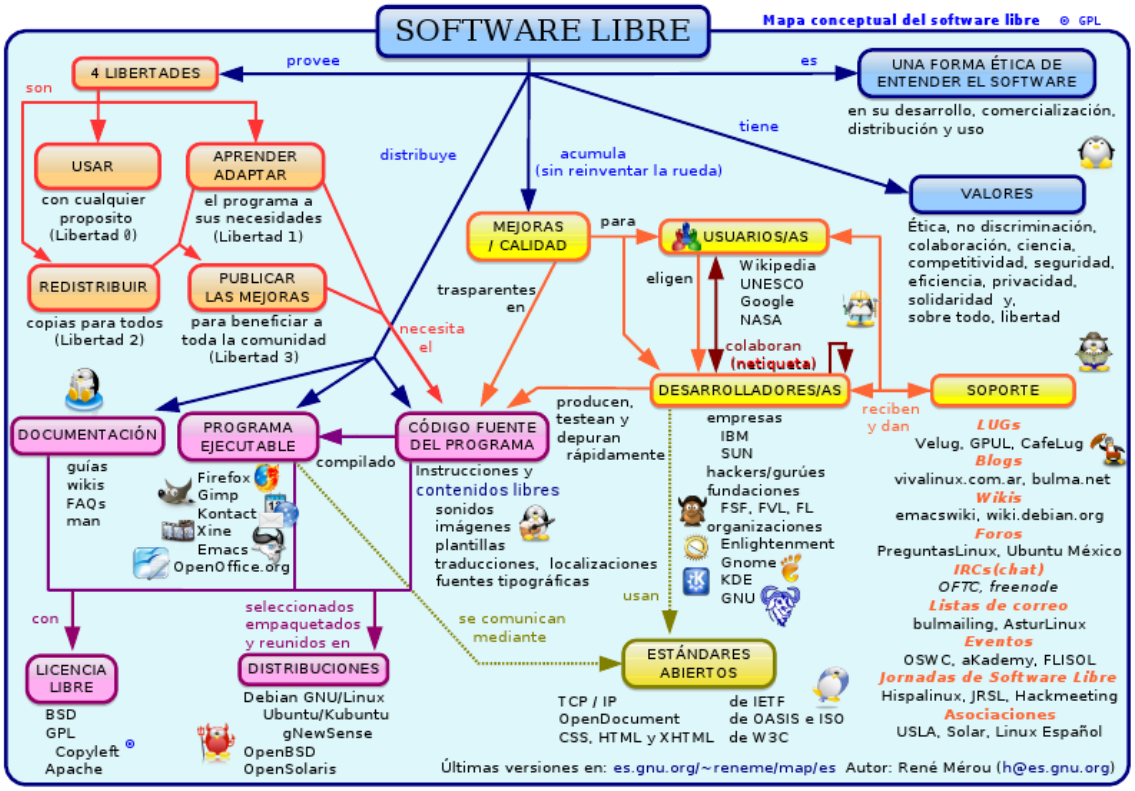


Figura 3: Mapa conceptual de Software Libre.

Fuente: https://es.wikipedia.org/wiki/Software_libre

2.2 SOFTWARE PROPIETARIO

El software propietario1 o privativo es el software del cual no existe una forma libre de acceso a su código fuente, el cual solo se encuentra a disposición de su desarrollador y no se permite su libre modificación, adaptación o incluso lectura por parte de terceros. Para la Fundación para el Software Libre (FSF), este concepto se aplica a cualquier programa informático que no es libre o que solo lo es parcialmente (semilibre), sea

[8] Wikipedia, la enciclopedia libre, (2021). *Software Libre.* Recuperado de https://es.wikipedia.org/wiki/Software_libre

porque su uso, redistribución o modificación está prohibida, o sea porque requiere permiso expreso del titular del software o software de aplicación.

La persona física o jurídica (compañía, corporación, fundación, etc.), al poseer los derechos de autor sobre un software, tiene la posibilidad de controlar y restringir los derechos del usuario sobre su programa, lo que en el software no libre implica por lo general que el usuario solo tendrá derecho a ejecutar el software bajo ciertas condiciones, comúnmente fijadas por el proveedor, que signifique la restricción de una o varias de las cuatro libertades.[9]

2.3 HARDWARE LIBRE

Se llama hardware libre, hardware de código abierto, electrónica libre o máquinas libres a aquellos dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago, o de forma gratuita. La filosofía del software libre es aplicable a la del hardware libre, y por eso forma parte de la cultura libre.

2.4 TARJETA DE DESARROLLO

Una tarjeta de desarrollo es una placa o circuito que contiene un microcontrolador principal que corre o ejecuta una serie de instrucciones de un programa suministrado. Alrededor de este procesador o unidad principal se ha creado un diseño electrónico que permite: la programación del componente, suministra el voltaje adecuado para el correcto funcionamiento del controlador y proporciona acceso a las entrada y salidas del microcontrolador para la conexión de sensores y actuadores.[10]

2.5 COMPUERTAS LÓGICAS

Las compuertas lógicas son circuitos electrónicos diseñados para obtener resultados booleanos (0,1), los cuales se obtienen de operaciones lógicas binarias (suma,

[9] Wikipedia, la enciclopedia libre, (2021). *Software Propietario.* Recuperado de https://es.wikipedia.org/wiki/Software_propietario

[10] rjconcepcion, (2021). *Tarjetas de desarrollo (Episodio #8).* Recuperado de https://www.rjconcepcion.com/podcast/tarjetas-de-desarrollo-episodio-8/

multiplicación). Dichas compuertas son AND, OR, NOT, NAND, NOR, XOR, XNOR. Además, se pueden conectar entre sí para obtener nuevas funciones. A continuación, se describirá las características de las compuertas. Este tipo de dispositivos lógicos se encuentran implementados con transistores y diodos en un semiconductor y actualmente podemos encontrarlas en formas de circuitos integrados lógicos. Al mismo tiempo, puedes tu programar el comportamiento de otra manera, con circuitos reconfigurables o programable, como microcontroladores o FPGA. Sin embargo, en este tutorial veremos las compuertas implementadas en circuitos independientes y su comportamiento.[11]

2.5.1 COMPUERTA NOT

En la compuerta NOT, el estado de la salida es inversa a la entrada. Evidentemente, una negación.

A	Q
0	**1**
1	**0**

Tabla 1: Tabla de verdad de la compuerta NOT.

Fuente: Diseño propio.

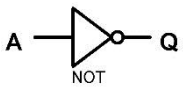


Figura 4: Compuerta NOT.

Fuente: Generado con Proteus 8.8

2.5.2 COMPUERTA OR

En la compuerta OR, el estado de su salida estará en 1, cuando cualquiera de sus entradas este en 1. De modo que es una suma lógica.

A	B	Q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Tabla 2: Tabla de verdad de la compuerta OR.

Fuente: Diseño propio.

[11] HETPRO, Herramientas Tecnológicas Profesionales, (2021). *Compuertas Lógicas.* Recuperado de https://hetpro-store.com/TUTORIALES/compuertas-logicas/

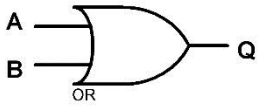


Figura 5: Compuerta OR.

Fuente: Generado con Proteus 8.8

2.5.3 COMPUERTA AND

En la compuerta AND, el estado de su salida estará en 1, solo cuando ambas entradas estén en 1. De modo que es una multiplicación lógica.

A	B	Q
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Tabla 3: Tabla de verdad de la compuerta AND.

Fuente: Diseño propio.

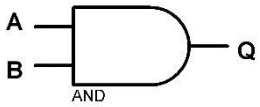


Figura 6: Compuerta AND.

Fuente: Generado con Proteus 8.8

2.5.4 COMPUERTA NOR

La compuerta NOR se compone por una compuerta OR y una compuerta NOT, por lo que la salida se invierte al ser negada.

A	B	Q
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Tabla 4: Tabla de verdad de la compuerta NOR.

Fuente: Diseño propio.

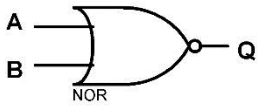


Figura 7: Compuerta NOR.

Fuente: Generado con Proteus 8.8

2.5.5 COMPUERTA NAND

En la compuerta NAND.

A	B	Q
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Tabla 5: Tabla de verdad de la compuerta NAND.

Fuente: Diseño propio.

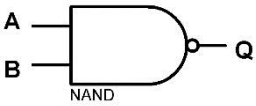


Figura 8: Compuerta NAND.

Fuente: Generado con Proteus 8.8

2.5.6 COMPUERTA XOR

En la compuerta XOR.

A	B	Q
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Tabla 6: Tabla de verdad de la compuerta XOR.

Fuente: Diseño propio.

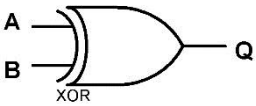


Figura 9: Compuerta XOR.

Fuente: Generado con Proteus 8.8

2.6 CODIFICADORES

Los codificadores son circuitos que codifican en forma binaria la información que se le aplica a su entrada, teniendo en cuenta que la información de entrada debe ser numérica o alfanumérica. Están compuestos por 2 ^ n entradas y n salidas.

En los codificadores, sólo una de las entradas puede estar activa. El código de salida indica qué entrada es la que está activa. Para denominar a los codificadores se puede emplear un sistema análogo al de los decodificadores, que estudiaremos más adelante, refiriéndose a su número de entradas y de salidas; así un decodificador que tuviera 4 entradas y 2 salidas sería un codificador 4:2.

Para evitar problemas cuando dos entradas del decodificador están activas se emplean los llamados codificadores con prioridad o codificadores prioritarios. En éstos cuando más de una entrada está activada será la de más peso la que determinará el código de salida.

Otro problema que puede presentarse es que para distintos valores de las entradas se tenga un mismo código de salida. Es lo que ocurre cuando no está activa la entrada de habilitación (si la hay) o, en caso de estar habilitado el codificador, no hay ninguna entrada binaria activa o lo está la menos significativa. En todos estos casos el valor de todas las salidas es cero.

Los codificadores nos permiten "compactar" la información, generando un código de salida a partir de la información de entrada.[12]

[12] BIRTLH, (2021). *ELEC02.- Circuitos combinacionales MSI.* Recuperado de https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_53_codificadores.html

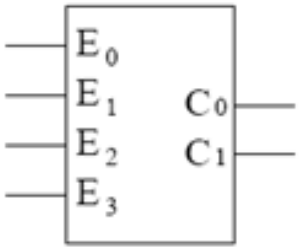


Figura 10: Codificador 4 a 2.

Fuente: https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_53_codificadores.html

E3	E2	E1	E0	C1	C0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Tabla 7: Tabla de verdad del codificador de 4 a 2.

Fuente: Diseño propio.

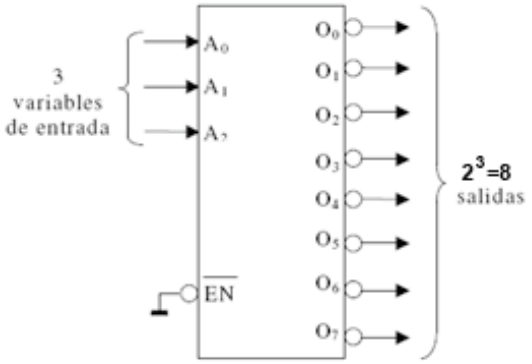
2.7 DECODIFICADORES

Un decodificador es un circuito combinacional, que, en su forma más general, posee n entradas y 2 ^ n salidas digitales, donde solamente una de las salidas puede estar activa permaneciendo el resto en reposo.

Cada combinación de las variables de entrada representa un número binario y activa una y sólo una de las salidas, aquella que corresponde al número decimal equivalente al código binario de entrada.

Una forma de denominar a los decodificadores es haciendo referencia a su número de entradas y salidas, por ejemplo, un decodificador de 2 entradas y 4 salidas es un decodificador 2:4, un decodificador de 3 entradas y 8 salidas es un decodificador 3:8. También se pueden denominar a los decodificadores haciendo referencia sólo al número de entradas.

Algunos de ellos están diseñados especialmente para la activación de visualización (display) del tipo de siete segmentos. Una aplicación muy popular, son los decodificadores excitadores de visualizadores o displays (decodificadores BCD a display de 7 segmentos).[13]



Entradas				Salidas							
EN	A_2	A_1	A_0	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
1	*	*	*	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

Figura 11: Decodificador 3 a 8.

Fuente: https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_54_decodificadores.html

2.8 MEMORIAS

En informática, la memoria es el dispositivo que retiene, memoriza o almacena datos informáticos durante algún periodo de tiempo.1La memoria proporciona una de las principales funciones de la computación moderna: el almacenamiento de información y conocimiento. Es uno de los componentes fundamentales de la computadora, que interconectada a la unidad central de procesamiento (CPU, por las siglas en inglés de Central Processing Unit) y los dispositivos de entrada/salida, implementan lo fundamental del modelo de computadora de la arquitectura de Von Neumann.

2.8.1 CARACTERÍSTICAS

La memoria volátil requiere energía constante para mantener la información almacenada. La memoria volátil se suele usar solo en memorias primarias. La memoria RAM es una memoria volátil, ya que pierde información en la falta de energía eléctrica.

[13] BIRTLH, (2021). *ELEC02.- Circuitos combinacionales MSI.* Recuperado de https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_54_decodificadores.html

La memoria no volátil retendrá la información almacenada incluso si no recibe corriente eléctrica constantemente, como es el caso de la memoria ROM. Se usa para almacenamientos a largo plazo y, por tanto, se usa en memorias secundarias, terciarias y fuera de línea.

La memoria dinámica es una memoria volátil que además requiere que periódicamente se refresque la información almacenada, o leída y reescrita sin modificaciones.[14]

2.9 MEMORIA RAM

La memoria de acceso aleatorio (Random Access Memory, RAM) se utiliza como memoria de trabajo de computadoras y otros dispositivos para el sistema operativo, los programas y la mayor parte del software. En la RAM se cargan todas las instrucciones que ejecuta la unidad central de procesamiento (CPU) y otras unidades del computador, además de contener los datos que manipulan los distintos programas.

Se denominan «de acceso aleatorio» porque se puede leer o escribir en una posición de memoria con un tiempo de espera igual para cualquier posición, no siendo necesario seguir un orden para acceder (acceso secuencial) a la información de la manera más rápida posible.

2.9.1 HISTORIA

Uno de los primeros tipos de memoria RAM fue la memoria de núcleo magnético, desarrollada entre 1949 y 1952 y usada en muchos computadores hasta el desarrollo de circuitos integrados a finales de los años 60 y principios de los 70. Esa memoria requería que cada bit estuviera almacenado en un toroide de material ferromagnético de algunos milímetros de diámetro, lo que resultaba en dispositivos con una capacidad de memoria muy pequeña. Antes que eso, las computadoras usaban relés y líneas de retardo de varios tipos construidas para implementar las funciones de memoria principal con o sin acceso aleatorio.

[14] Wikipedia, la enciclopedia libre, (2021). *Memoria (informática).* Recuperado de https://es.wikipedia.org/wiki/Memoria_(inform%C3%A1tica)

En 1969 fueron lanzadas una de las primeras memorias RAM basadas en semiconductores de silicio por parte de Intel con el integrado 3101 de 64 bits de memoria y para el siguiente año se presentó una memoria DRAM de 1024 bits, referencia 1103 que se constituyó en un hito, ya que fue la primera en ser comercializada con éxito, lo que significó el principio del fin para las memorias de núcleo magnético. En comparación con los integrados de memoria DRAM actuales, la 1103 es primitiva en varios aspectos, pero tenía un desempeño mayor que la memoria de núcleos.

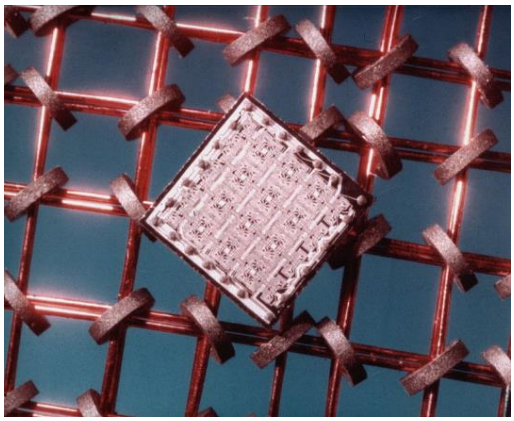


Figura 12: Integrado de silicio de 64 bits sobre un sector de memoria de núcleo magnético.
Fuente: https://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio

En 1973 se presentó una innovación que permitió otra miniaturización y se convirtió en estándar para las memorias DRAM: la multiplexación en tiempo de las direcciones de memoria. MOSTEK lanzó la referencia MK4096 de 4096 bytes en un empaque de 16 pines, mientras sus competidores las fabricaban en el empaque DIP de 22 pines. El esquema de direccionamiento2 se convirtió en un estándar de facto debido a la gran popularidad que logró esta referencia de DRAM. Para finales de los 70 los integrados eran usados en la mayoría de computadores nuevos, se soldaban directamente a las placas base o se instalaban en zócalos, de manera que ocupaban un área extensa de circuito impreso. Con el tiempo se hizo obvio que la instalación de RAM sobre el impreso principal, impedía la miniaturización, entonces se idearon los primeros módulos de memoria como el SIPP, aprovechando las ventajas de la construcción modular. El formato SIMM fue una mejora al anterior, eliminando los pines metálicos y dejando unas áreas de cobre en uno de los bordes del impreso, muy similares a los de las tarjetas

de expansión, de hecho, los módulos SIPP y los primeros SIMM tienen la misma distribución de pines.

2.9.2 TIPOS DE RAM

Las dos formas principales de RAM moderna son:

1. SRAM (Static Random Access Memory), RAM estática, memoria estática de acceso aleatorio.
    - volátiles.
    - no volátiles:
2. NVRAM (non-volatile random access memory), memoria de acceso aleatorio no volátil
3. MRAM (magnetoresistive random-access memory), memoria de acceso aleatorio magnetorresistiva o magnética
4. DRAM (Dynamic Random Access Memory), RAM dinámica, memoria dinámica de acceso aleatorio.
    - DRAM Asincrónica (Asynchronous Dynamic Random Access Memory), memoria de acceso aleatorio dinámica asincrónica.
        - FPM RAM (Fast Page Mode RAM)
        - EDO RAM (Extended Data Output RAM)
5. SDRAM (Synchronous Dynamic Random-Access Memory, memoria de acceso aleatorio dinámica sincrónica)
    - Rambus:
        - RDRAM (Rambus Dynamic Random Access Memory)
        - XDR DRAM (eXtreme Data Rate Dynamic Random Access Memory)
        - XDR2 DRAM (eXtreme Data Rate two Dynamic Random Access Memory)
    - SDR SDRAM (Single Data Rate Synchronous Dynamic Random-Access Memory, SDRAM de tasa de datos simple)

- DDR SDRAM (Double Data Rate Synchronous Dynamic Random-Access Memory, SDRAM de tasa de datos doble)
- DDR2 SDRAM (Double Data Rate type two SDRAM, SDRAM de tasa de datos doble de tipo dos)
- DDR3 SDRAM (Double Data Rate type three SDRAM, SDRAM de tasa de datos doble de tipo tres)
- DDR4 SDRAM (Double Data Rate type four SDRAM, SDRAM de tasa de datos doble de tipo cuatro).
- DDR5 SDRAM (Double Data Rate type five SDRAM, SDRAM de tasa de datos doble de tipo cinco).
- DDR6 SDRAM (Double Data Rate type six SDRAM, SDRAM de tasa de datos doble de tipo seis).[15]

2.10 MEMORIA ROM

La memoria de solo lectura, conocida también como ROM (acrónimo en inglés de read-only memory), es un medio de almacenamiento utilizado en ordenadores y dispositivos electrónicos, que permite solo la lectura de la información y no su escritura,1 independientemente de la presencia o no de una fuente de energía.

Los datos almacenados en la ROM no se pueden modificar, o al menos no de manera rápida o fácil. Se utiliza principalmente para contener el firmware (programa que está estrechamente ligado a hardware específico, y es poco probable que requiera actualizaciones frecuentes) u otro contenido vital para el funcionamiento del dispositivo, como los programas que ponen en marcha el ordenador y realizan los diagnósticos.

En su sentido más estricto, se refiere solo a máscara ROM -en inglés, MROM- (el más antiguo tipo de estado sólido ROM), que se fabrica con los datos almacenados de forma permanente, y por lo tanto, su contenido no puede ser modificado de ninguna forma. Sin embargo, las ROM más modernas, como EPROM y Flash EEPROM, efectivamente se pueden borrar y volver a programar varias veces, aun siendo descritos como "memoria

[15] Wikipedia, la enciclopedia libre, (2021). *Memoria de acceso aleatorio*. Recuperado de https://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio

de solo lectura" (ROM). La razón de que se las continúe llamando así es que el proceso de reprogramación en general es poco frecuente, relativamente lento y, a menudo, no se permite la escritura en lugares aleatorios de la memoria.

2.10.1 HISTORIA

El tipo más simple de ROM en estado sólido es de la misma antigüedad que la propia tecnología semiconductora. Las puertas lógicas combinacionales pueden usarse en conjunto para indexar una dirección de memoria de n bits en valores de m bits de tamaño (una tabla de consultas). Con la invención de los circuitos integrados se desarrolló la máscara ROM. La máscara ROM consistía en una cuadrícula de líneas formadas por una palabra y líneas formadas por un bit seleccionadas respectivamente a partir de cambios en el transistor. De esta manera podían representar una tabla de consultas arbitraria y un lapso de propagación deducible.

En las máscaras ROM los datos están físicamente codificados en el mismo circuito, así que solo se pueden programar durante la fabricación. Esto acarrea serias desventajas:

- Solo es económico comprarlas en grandes cantidades, ya que el usuario contrata fundiciones para producirlas según sus necesidades.
- El tiempo transcurrido entre completar el diseño de la máscara y recibir el resultado final es muy largo.
- No son prácticas para I+D por el hecho de que los desarrolladores necesitan cambiar el contenido de la memoria mientras refinan un diseño.
- Si un producto tiene un error en la máscara, la única manera de arreglarlo es reemplazando físicamente la ROM por otra.

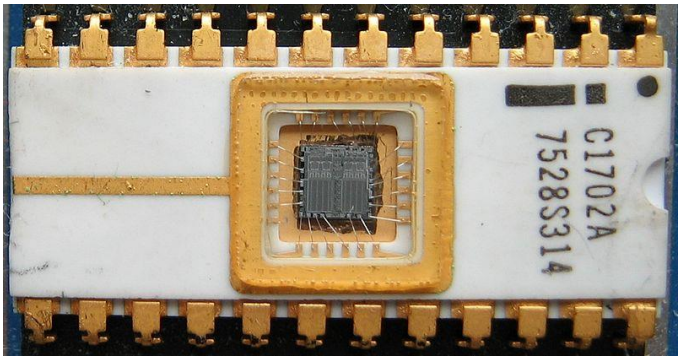


Figura 13: Primera eeprom Intel 1702.

Fuente: https://es.wikipedia.org/wiki/Memoria_de_solo_lectura

Los desarrollos posteriores tomaron en cuenta estas deficiencias, así pues, se creó la memoria de solo lectura programable (PROM). Inventada en 1956, permitía a los usuarios modificarla solo una vez, alterando físicamente su estructura con la aplicación de pulsos de alto voltaje. Esto eliminó los problemas 1 y 2 antes mencionados, ya que una compañía podía pedir un gran lote de PROMs vacías y programarlas con el contenido necesario elegido por los diseñadores. En 1971 se desarrolló la memoria de solo lectura programable y borrable (EPROM) que permitía reiniciar su contenido exponiendo el dispositivo a fuertes rayos ultravioleta. De esta manera erradicaba el punto 3 de la anterior lista. Más tarde, en 1983, se inventó la EEPROM, resolviendo el conflicto número 4 de la lista ya que se podía reprogramar el contenido mientras proveyese un mecanismo para recibir contenido externo (por ejemplo, a través de un cable serial). En medio de la década de 1980 Toshiba inventó la memoria flash, una forma de EEPROM que permitía eliminar y reprogramar contenido en una misma operación mediante pulsos eléctricos miles de veces sin sufrir ningún daño.

Todas estas tecnologías mejoraron la versatilidad y flexibilidad de la ROM, pero lo hicieron a expensas de un alto incremento del costo por chip. Por eso las máscaras ROM se mantuvieron como la solución económica durante bastante tiempo. Esto fue así aproximadamente hasta el año 2000, cuando el precio de las memorias reprogramables hubo descendido lo suficiente como para comenzar a desplazar a las ROM no reprogramables del mercado.

El producto más reciente es la memoria NAND, otra vez desarrollada por Toshiba. Los diseñadores rompieron explícitamente con las prácticas del pasado, afirmando que

enfocaba "ser un reemplazo de los discos duros", más que tener el tradicional uso de la ROM como una forma de almacenamiento primario no volátil. En 2007, NAND ha avanzado bastante en su meta, ofreciendo un rendimiento comparable al de los discos duros, una mejor tolerancia a los shocks físicos, una miniaturización extrema (como por ejemplo memorias USB y tarjetas de memoria MicroSD), y un consumo de potencia mucho más bajo.

Los ordenadores domésticos a comienzos de los años 1980 venían con todo su sistema operativo en ROM. No había otra alternativa razonable ya que las unidades de disco eran generalmente opcionales. La actualización a una nueva versión significa usar un soldador o un grupo de interruptores DIP y reemplazar el viejo chip de ROM por uno nuevo. Actualmente los sistemas operativos en general ya no van en ROM. Todavía los ordenadores pueden dejar algunos de sus programas en memoria ROM, pero incluso en este caso, es más frecuente que vaya en memoria flash. Los teléfonos móviles y los asistentes personales digitales (PDA) suelen tener programas en memoria ROM (o por lo menos en memoria flash).

Algunas de las videoconsolas que usan programas basados en la memoria ROM son la Super Nintendo, la Nintendo 64, la Sega Mega Drive o la Game Boy. Estas memorias ROM, pegadas a cajas de plástico aptas para ser utilizadas e introducidas repetidas veces, son conocidas como cartuchos. Por extensión la palabra ROM puede referirse también a un archivo de datos que contenga una imagen del programa que se distribuye normalmente en memoria ROM, como una copia de un cartucho de videojuego.

Figura 14: Memoria de solo lectura conteniendo el BIOS de una vieja placa madre.
Fuente: https://es.wikipedia.org/wiki/Memoria_de_solo_lectura

Como la ROM no puede ser modificada (al menos en la antigua versión de máscara), solo resulta apropiada para almacenar datos que no necesiten ser modificados durante la vida de este dispositivo. Con este fin, la ROM se ha utilizado en muchos ordenadores para guardar tablas de consulta, utilizadas para la evaluación de funciones matemáticas y lógicas. Esto era especialmente eficiente cuando la unidad central de procesamiento era lenta y la ROM era barata en comparación con la RAM. De hecho, una razón de que todavía se utilice la memoria ROM para almacenar datos es la velocidad, ya que los discos siguen siendo más lentos. Y lo que es aún más importante, no se puede leer un programa que es necesario para ejecutar un disco desde el propio disco. Por lo tanto, la BIOS, o el sistema de arranque oportuno del PC normalmente se encuentran en una memoria ROM.

No obstante, el uso de la ROM para almacenar grandes cantidades de datos ha ido desapareciendo casi completamente en los ordenadores de propósito general, mientras que la memoria Flash ha ido ocupando este puesto.

2.10.2 TIPOS DE ROM

Los chips de la máscara programada ROM clásica son circuitos integrados que codifican físicamente los datos a almacenar, y por lo tanto es imposible cambiar su contenido después de la fabricación. Otros tipos de memoria de estado sólido no volátil permiten algún grado de modificación:

La memoria de solo lectura programable (PROM), o la ROM programable una sola vez (OTP), pueden ser escritas o programadas a través de un dispositivo especial llamado un programador PROM. Normalmente, este dispositivo utiliza alto voltaje para destruir o crear permanentemente enlaces internos (fusibles o antifusibles) dentro del chip. En consecuencia, una PROM solo puede programarse una vez.

La memoria de solo lectura programable y borrable (EPROM) puede ser borrada por la exposición a una fuerte luz ultravioleta (en general durante 10 minutos o más), a continuación, se reescribe otra vez con un proceso que necesita un voltaje más alto que el habitual aplicado. La exposición repetida a la luz UV desgastará eventualmente una EPROM, pero la resistencia de la mayoría de los chips EPROM excede 1000 ciclos de

borrado y reprogramación. Después de la programación, la ventana se cubre normalmente con una etiqueta para evitar el borrado accidental. Algunos chips EPROM son borrados de fábrica antes de ser empaquetados, y no incluyen ninguna ventana; estos son efectivamente PROM.

La memoria de solo lectura eléctricamente programable y borrable (EEPROM) se basa en una estructura de semiconductor similar a la EPROM, pero permite que todo su contenido (o bancos seleccionados) sea borrado eléctricamente, a continuación, reescrito eléctricamente, por lo que no deben ser retirados del ordenador (o una cámara, reproductor MP3, etc.). Escribir o flashear una EEPROM es mucho más lento (milisegundos por bit) que leer de una ROM o escribir a una RAM (nanosegundos en ambos casos). Existen diferentes tipos de EEPROM:

La memoria de solo lectura eléctricamente alterable (EAROM) es un tipo de EEPROM que se puede modificar un bit cada vez. La escritura es un proceso muy lento y necesita de nuevo un voltaje más alto (generalmente alrededor de 12 V) del que se utiliza para el acceso de lectura. EAROMs están destinados para aplicaciones que requieren reescritura poco frecuente y sólo parcial. EAROM puede ser utilizado como almacenamiento no volátil para obtener información de configuración del sistema crítico; en muchas aplicaciones, EAROM ha sido suplantada por la RAM CMOS suministrada por la red eléctrica y apoyada con una batería de litio.

La memoria flash (o simplemente flash) es un tipo moderno de EEPROM inventado en 1984. La memoria flash se puede borrar y volver a escribir más rápidamente que la EEPROM ordinaria, y los nuevos diseños cuentan con muy alta resistencia (superior a 1.000.000 de ciclos). La Flash NAND moderna hace uso eficiente de área de chip de silicio, lo que resulta en circuitos integrados individuales con una capacidad de hasta 32 GB a partir de 2007; esta característica, junto con su resistencia y durabilidad física, ha permitido la flash NAND reemplazar magnético en algunas aplicaciones (como las unidades flash USB). La memoria flash es a veces llamado flash ROM o Flash EEPROM cuando se usa como un reemplazo para los tipos de ROM viejos, pero no en

aplicaciones que aprovechan su capacidad de ser modificado rápidamente y con frecuencia.[16]

2.11 MICROPROCESADOR

El microprocesador (o simplemente procesador) es el circuito integrado central más complejo de un sistema informático; a modo de ilustración, se le suele llamar por analogía el «cerebro» de un ordenador.

Es el encargado de ejecutar todos los programas, desde el sistema operativo hasta las aplicaciones de usuario; solo ejecuta instrucciones en lenguaje binario, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir, las lógicas binarias y accesos a memoria.

Puede contener una o más unidades centrales de procesamiento (CPU) constituidas, esencialmente, por registros, una unidad de control, una unidad aritmético lógica (ALU) y una unidad de cálculo en coma flotante (conocida antiguamente como «coprocesador matemático»).

2.11.1 HISTORIA

El microprocesador surgió de la evolución de distintas tecnologías predecesoras, básicamente de la computación y de la tecnología de semiconductores. El inicio de esta última data de mitad de la década de 1950; estas tecnologías se fusionaron a principios de los años 1970, produciendo el primer microprocesador. Dichas tecnologías iniciaron su desarrollo a partir de la segunda guerra mundial; en este tiempo los científicos desarrollaron computadoras específicas para aplicaciones militares. En la posguerra, a mediados de la década de 1940, la computación digital emprendió un fuerte crecimiento también para propósitos científicos y civiles. La tecnología electrónica avanzó y los científicos hicieron grandes progresos en el diseño de componentes de estado sólido (semiconductores). En 1948 en los laboratorios Bell crearon el transistor.

[16] Wikipedia, la enciclopedia libre, (2021). *Memoria de solo lectura*. Recuperado de https://es.wikipedia.org/wiki/Memoria_de_solo_lectura

En los años 1950, aparecieron las primeras computadoras digitales de propósito general. Se fabricaron utilizando tubos al vacío o bulbos como componentes electrónicos activos. Módulos de tubos al vacío componían circuitos lógicos básicos, tales como compuertas y flip-flops. Ensamblándolos en módulos se construyó la computadora electrónica (la lógica de control, circuitos de memoria, etc.). Los tubos de vacío también formaron parte de la construcción de máquinas para la comunicación con las computadoras.

Para la construcción de un circuito sumador simple se requiere de algunas compuertas lógicas. La construcción de una computadora digital precisa numerosos circuitos o dispositivos electrónicos. Un paso trascendental en el diseño de la computadora fue hacer que el dato fuera almacenado en memoria. Y la idea de almacenar programas en memoria para luego ejecutarlo fue también de fundamental importancia (Arquitectura de von Neumann).

La tecnología de los circuitos de estado sólido evolucionó en la década de 1950. El empleo del silicio (Si), de bajo costo y con métodos de producción masiva, hicieron del transistor el componente más usado para el diseño de circuitos electrónicos. Por lo tanto el diseño de la computadora digital se reemplazó del tubo al vacío por el transistor, a finales de la década de 1950.

A principios de la década de 1960, el estado de arte en la construcción de computadoras de estado sólido sufrió un notable avance; surgieron las tecnologías en circuitos digitales como: RTL (Lógica Transistor Resistor), DTL (Lógica Transistor Diodo), TTL (Lógica Transistor Transistor), ECL (Lógica Complementada Emisor).

A mediados de los años 1960 se producen las familias de circuitos de lógica digital, dispositivos integrados en escala SSI y MSI que corresponden a baja y mediana escala de integración de componentes. A finales de los años 1960 y principios de los 70 surgieron los sistemas a alta escala de integración o LSI. La tecnología LSI fue haciendo posible incrementar la cantidad de componentes en los circuitos integrados. Sin embargo, pocos circuitos LSI fueron producidos, los dispositivos de memoria eran un buen ejemplo.

Las primeras calculadoras electrónicas requerían entre 75 y 100 circuitos integrados. Después se dio un paso importante en la reducción de la arquitectura de la computadora a un circuito integrado simple, resultando uno que fue llamado microprocesador, unión de las palabras «Micro» del griego μικρο-, «pequeño», y procesador. Sin embargo, es totalmente válido usar el término genérico procesador, dado que, con el paso de los años, la escala de integración se ha visto reducida de micro métrica a nanométrica; y, además, es, sin duda, un procesador.

- El primer microprocesador fue el Intel 4004 de Intel Corporation, producido en 1971. Se desarrolló originalmente para una calculadora y resultó revolucionario para su época. Contenía 2300 transistores, era un microprocesador de arquitectura de 4 bits que podía realizar hasta 60 000 operaciones por segundo trabajando a una frecuencia de reloj de alrededor de 700 kHz.
- El primer microprocesador de 8 bits fue el Intel 8008, desarrollado a mediados de 1972 para su uso en terminales informáticos. El Intel 8008 integraba 3300 transistores y podía procesar a frecuencias máximas de 800 kHz.
- El primer microprocesador realmente diseñado para uso general, desarrollado en 1974, fue el Intel 8080 de 8 bits, que contenía 4500 transistores y podía ejecutar 200 000 instrucciones por segundo trabajando a alrededor de 2 MHz.
- El primer microprocesador de 16 bits fue el 8086, seguido del 8088. El 8086 fue el inicio y el primer miembro de la popular arquitectura x86, actualmente usada en la mayoría de los computadores. El chip 8086 fue introducido al mercado en el verano de 1978, pero debido a que no había aplicaciones en el mercado que funcionaran con 16 bits, Intel sacó al mercado el 8088, que fue lanzado en 1979. Llegaron a operar a frecuencias mayores de 4 MHz.
- El microprocesador elegido para equipar al IBM Personal Computer/AT, que causó que fuera el más empleado en los PC-AT compatibles entre mediados y finales de los años 1980 fue el Intel 80286 (también conocido simplemente como 286); es un microprocesador de 16 bits, de la familia x86, que fue lanzado al mercado en 1982. Contaba con 134 000 transistores. Las versiones finales alcanzaron velocidades de hasta 25 MHz.

- Uno de los primeros procesadores de arquitectura de 32 bits fue el 80386 de Intel, fabricado a mediados y fines de la década de 1980; en sus diferentes versiones llegó a trabajar a frecuencias del orden de los 40 MHz.
- El microprocesador DEC Alpha se lanzó al mercado en 1992, corriendo a 200 MHz en su primera versión, en tanto que el Intel Pentium surgió en 1993 con una frecuencia de trabajo de 66 MHz. El procesador Alpha, de tecnología RISC y arquitectura de 64 bits, marcó un hito, declarándose como el más rápido del mundo, en su época. Llegó a 1 GHz de frecuencia hacia el año 2001. Irónicamente, a mediados del 2003, cuando se pensaba quitarlo de circulación, el Alpha aún encabezaba la lista de los microprocesadores más rápidos de Estados Unidos.
- Los microprocesadores modernos tienen una capacidad y velocidad mucho mayores, trabajan en arquitecturas de 64 bits, integran más de 700 millones de transistores, como es en el caso de la serie Core i7, y pueden operar a frecuencias normales algo superiores a los 3 GHz (3000 MHz).

2.11.2 EVOLUCIÓN

Hasta los primeros años de la década de 1970 los diferentes componentes electrónicos que formaban un procesador no podían ser un único circuito integrado, era necesario utilizar dos o tres "chips" para hacer una CPU (uno era el "ALU" - Arithmetical Logic Unit, el otro la " control Unit", el otro el " Register Bank", etc.). En 1971 la compañía Intel consiguió por primera vez poner todos los transistores que constituían un procesador sobre un único circuito integrado, el 4004, nacía el microprocesador.

Seguidamente se expone una lista ordenada cronológicamente de los microprocesadores más populares que fueron surgiendo. En la URSS se realizaron otros sistemas que dieron lugar a la serie microprocesador Elbrus.

1971: El Intel 4004

El 4004 fue el primer microprocesador del mundo, creado en un simple chip y desarrollado por Intel. Era un CPU de 4 bits y también fue el primero disponible comercialmente. Este desarrollo impulsó la calculadora de Busicom e inició el camino

para dotar de «inteligencia» a objetos inanimados y, asimismo, a la computadora personal.

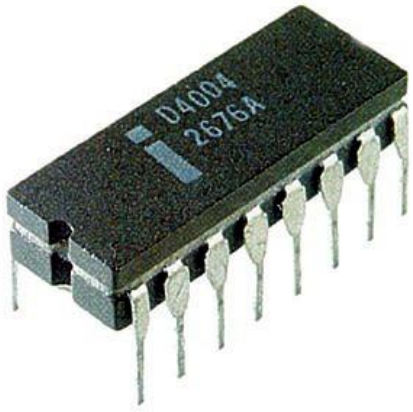


Figura 15: Microprocesador 4004 de Intel.
Fuente: https://commons.wikimedia.org/wiki/File:Intel_4004.jpg

1972: El Intel 8008

Codificado inicialmente como 1201, fue pedido a Intel por Computer Terminal Corporation para usarlo en su terminal programable Datapoint 2200, pero debido a que Intel terminó el proyecto tarde y a que no cumplía con las expectativas de Computer Terminal Corporation, finalmente no fue usado en el Datapoint. Posteriormente Computer Terminal Corporation e Intel acordaron que el i8008 pudiera ser vendido a otros clientes.

1974: El SC/MP

El SC/MP desarrollado por National Semiconductor, fue uno de los primeros microprocesadores, y estuvo disponible desde principio de 1974. El nombre SC/MP (popularmente conocido como «Scamp») es el acrónimo de Simple Cost-effective Micro Processor (Microprocesador simple y rentable). Presenta un bus de direcciones de 16 bits y un bus de datos de 8 bits. Una característica, avanzada para su tiempo, es la capacidad de liberar los buses a fin de que puedan ser compartidos por varios procesadores. Este microprocesador fue muy utilizado, por su bajo costo, y provisto en kits, para propósitos educativos, de investigación y para el desarrollo de controladores industriales diversos.

1974: El Intel 8080

EL 8080 se convirtió en la CPU de la primera computadora personal, la Altair 8800 de MITS, según se alega, nombrada así por un destino de la Nave Espacial «Starship» del programa de televisión Viaje a las Estrellas, y el IMSAI 8080, formando la base para las máquinas que ejecutaban el sistema operativo CP/M-80. Los fanáticos de las computadoras podían comprar un equipo Altair por un precio (en aquel momento) de 395 USD. En un periodo de pocos meses, se vendieron decenas de miles de estos PC.

1975: Motorola 6800

Se fabrica, por parte de Motorola, el Motorola MC6800, más conocido como 6800. Su nombre proviene de que contenía aproximadamente 6800 transistores.4 Fue lanzado al mercado poco después del Intel 8080.5 Varias de las primeras microcomputadoras de los años 1970 usaron el 6800 como procesador. Entre ellas se encuentran la SWTPC 6800, que fue la primera en usarlo, y la muy conocida Altair 680. Este microprocesador se utilizó profusamente como parte de un kit para el desarrollo de sistemas controladores en la industria. Partiendo del 6800 se crearon varios procesadores derivados, siendo uno de los más potentes el Motorola 6809

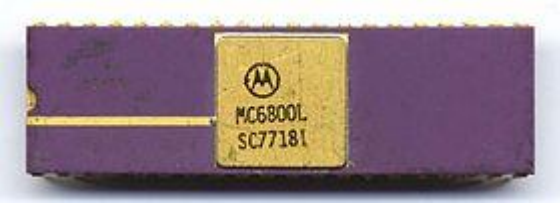


Figura 16: Microprocesador MC6800 de Motorola.
Fuente: https://commons.wikimedia.org/wiki/File:Motorola_MC6800L_SC7718I_top.jpg

1976: El Z80

La compañía Zilog Inc. crea el Zilog Z80. Es un microprocesador de 8 bits construido en tecnología NMOS, y fue basado en el Intel 8080. Básicamente es una ampliación de este, con lo que admite todas sus instrucciones. Un año después sale al mercado el primer computador que hace uso del Z80, el Tandy TRS-80 Model 1 provisto de un Z80 a 1,77 MHz y 4 KB de RAM. Es uno de los procesadores de más éxito del mercado, del cual se han producido numerosas versiones clónicas, y sigue siendo usado de forma

extensiva en la actualidad en multitud de sistemas embebidos. La compañía Zilog fue fundada 1974 por Federico Faggin, quien fue diseñador jefe del microprocesador Intel 4004 y posteriormente del Intel 8080.

Figura 17: Microprocesador Z80 de Zilog.

Fuente: https://commons.wikimedia.org/wiki/File:Ic-photo-zilog-Z0840008PSC-Z80-CPU.png

1978: Los Intel 8086 y 8088

Una venta realizada por Intel a la nueva división de computadoras personales de IBM, hizo que las PC de IBM dieran un gran golpe comercial con el nuevo producto con el 8088, el llamado IBM PC. El éxito del 8088 propulsó a Intel a la lista de las 500 mejores compañías, en la prestigiosa revista Fortune, y la misma nombró la empresa como uno de Los triunfos comerciales de los sesenta.

1982: El Intel 80286

El 80286, popularmente conocido como 286, fue el primer procesador de Intel que podría ejecutar todo el software escrito para su predecesor. Esta compatibilidad del software sigue siendo un sello de la familia de microprocesadores de Intel. Luego de seis años de su introducción, había un estimado de 15 millones de PC basadas en el 286, instaladas alrededor del mundo.

Figura 18: Microprocesador 80286 de Intel.

Fuente: https://commons.wikimedia.org/wiki/File:Intel_80286_68pin_plastic_10mhz_2007_03_27.jpg

1985: El Intel 80386

Este procesador Intel, popularmente llamado 386, se integró con 275 000 transistores, más de 100 veces tantos como en el original 4004. El 386 añadió una arquitectura de 32 bits, con capacidad para multitarea y una unidad de traslación de páginas, lo que hizo mucho más sencillo implementar sistemas operativos que usaran memoria virtual.

1985: El VAX 78032

El microprocesador VAX 78032 (también conocido como DC333), es de único chip y de 32 bits, y fue desarrollado y fabricado por Digital Equipment Corporation (DEC); instalado en los equipos MicroVAX II, en conjunto con su chip coprocesador de coma flotante separado, el 78132, tenían una potencia cercana al 90 % de la que podía entregar el minicomputador VAX 11/780 que fuera presentado en 1977. Este microprocesador contenía 125 000 transistores, fue fabricado con la tecnología ZMOS de DEC. Los sistemas VAX y los basados en este procesador fueron los preferidos por la comunidad científica y de ingeniería durante la década de 1980.

1989: El Intel 80486

La generación 486 realmente significó contar con una computadora personal de prestaciones avanzadas, entre ellas, un conjunto de instrucciones optimizado, una unidad de coma flotante o FPU, una unidad de interfaz de bus mejorada y una memoria caché unificada, todo ello integrado en el propio chip del microprocesador. Estas mejoras hicieron que los i486 fueran el doble de rápidos que el par i386-i387 operando a la misma frecuencia de reloj. El procesador Intel 486 fue el primero en ofrecer un coprocesador matemático o FPU integrado; con él que se aceleraron notablemente las operaciones de cálculo. Usando una unidad FPU las operaciones matemáticas más complejas son realizadas por el coprocesador de manera prácticamente independiente a la función del procesador principal.

1991: El AMD AMx86

Procesadores fabricados por AMD 100 % compatible con los códigos de Intel de ese momento. Llamados «clones» de Intel, llegaron incluso a superar la frecuencia de reloj

de los procesadores de Intel y a precios significativamente menores. Aquí se incluyen las series Am286, Am386, Am486 y Am586.

1993: El Intel Pentium

El microprocesador de Pentium poseía una arquitectura capaz de ejecutar dos operaciones a la vez, gracias a sus dos tuberías de datos de 32 bits cada uno, uno equivalente al 486DX(u) y el otro equivalente a 486SX(u). Además, estaba dotado de un bus de datos de 64 bits, y permitía un acceso a memoria de 64 bits (aunque el procesador seguía manteniendo compatibilidad de 32 bits para las operaciones internas, y los registros también eran de 32 bits). Las versiones que incluían instrucciones MMX no solo brindaban al usuario un más eficiente manejo de aplicaciones multimedia, sino que también se ofrecían en velocidades de hasta 233 MHz. Se incluyó una versión de 200 MHz y la más básica trabajaba a alrededor de 166 MHz de frecuencia de reloj. El nombre Pentium, se mencionó en las historietas y en charlas de la televisión a diario, en realidad se volvió una palabra muy popular poco después de su introducción.

1994: EL PowerPC 620

En este año IBM y Motorola desarrollan el primer prototipo del procesador PowerPC de 64 bit,6 la implementación más avanzada de la arquitectura PowerPC, que estuvo disponible al año próximo. El 620 fue diseñado para su utilización en servidores, y especialmente optimizado para usarlo en configuraciones de cuatro y hasta ocho procesadores en servidores de aplicaciones de base de datos y vídeo. Este procesador incorpora siete millones de transistores y corre a 133 MHz. Es ofrecido como un puente de migración para aquellos usuarios que quieren utilizar aplicaciones de 64 bits, sin tener que renunciar a ejecutar aplicaciones de 32 bits.

1995: EL Intel Pentium Pro

Lanzado al mercado en otoño de 1995, el procesador Pentium Pro (profesional) se diseñó con una arquitectura de 32 bits. Se usó en servidores y los programas y aplicaciones para estaciones de trabajo (de redes) impulsaron rápidamente su integración en las computadoras. El rendimiento del código de 32 bits era excelente, pero el Pentium

Pro a menudo era más lento que un Pentium cuando ejecutaba código o sistemas operativos de 16 bits. El procesador Pentium Pro estaba compuesto por alrededor de 5'5 millones de transistores.

2000: EL Intel Pentium 4

Este es un microprocesador de séptima generación basado en la arquitectura x86 y fabricado por Intel. Es el primero con un diseño completamente nuevo desde el Pentium Pro. Se estrenó la arquitectura NetBurst, la cual no daba mejoras considerables respecto a la anterior P6. Intel sacrificó el rendimiento de cada ciclo para obtener a cambio mayor cantidad de ciclos por segundo y una mejora en las instrucciones SSE.

2006: El Intel Core Duo

Intel lanzó esta gama de procesadores de doble núcleo y CPUs 2x2 MCM (módulo Multi-Chip) de cuatro núcleos con el conjunto de instrucciones x86-64, basado en la nueva arquitectura Core de Intel. La microarquitectura Core regresó a velocidades de CPU bajas y mejoró el uso del procesador de ambos ciclos de velocidad y energía comparados con anteriores NetBurst de los CPU Pentium 4/D2. La microarquitectura Core provee etapas de decodificación, unidades de ejecución, caché y buses más eficientes, reduciendo el consumo de energía de CPU Core 2, mientras se incrementa la capacidad de procesamiento. Los CPU de Intel han variado muy bruscamente en consumo de energía de acuerdo a velocidad de procesador, arquitectura y procesos de semiconductor, mostrado en las tablas de disipación de energía del CPU. Esta gama de procesadores fueron fabricados de 65 a 45 nanómetros.

2008: El Intel Core i7 Nehalem

Intel Core i7 es una familia de procesadores de cuatro núcleos de la arquitectura Intel x86-64. Los Core i7 son los primeros procesadores que usan la microarquitectura Nehalem de Intel y es el sucesor de la familia Intel Core 2. FSB es reemplazado por la interfaz QuickPath en i7 e i5 (zócalo 1366), y sustituido a su vez en i7, i5 e i3 (zócalo 1156) por el DMI eliminado el northBrige e implementando puertos PCI Express directamente. Memoria de tres canales (ancho de datos de 192 bits): cada canal puede

soportar una o dos memorias DIMM DDR3. La placa base compatible con Core i7 tiene cuatro (3+1) o seis ranuras DIMM en lugar de dos o cuatro, y las DIMM deben ser instaladas en grupos de tres, no dos. El Hyperthreading fue reimplementado creando núcleos lógicos. Está fabricado a arquitecturas de 45 nm y 32 nm y posee 731 millones de transistores su versión más potente. Se volvió a usar frecuencias altas, aunque a contrapartida los consumos se dispararon.

2011: El Intel Core Sandy Bridge

Llegan para reemplazar los chips Nehalem, con Intel Core i3, Intel Core i5 e Intel Core i7 serie 2000 y Pentium G.

Intel lanzó sus procesadores que se conocen con el nombre en clave Sandy Bridge. Estos procesadores Intel Core que no tienen sustanciales cambios en arquitectura respecto a nehalem, pero si los necesarios para hacerlos más eficientes y rápidos que los modelos anteriores. Es la segunda generación de los Intel Core con nuevas instrucciones de 256 bits, duplicando el rendimiento, mejorando el rendimiento en 3D y todo lo que se relacione con operación en multimedia. Llegaron la primera semana de enero del 2011. Incluye nuevo conjunto de instrucciones denominado AVX y una GPU integrada de hasta 12 unidades de ejecución

2013: El Intel Core Haswell

Haswell es el nombre clave de los procesadores de cuarta generación de Intel Core. Son la corrección de errores de la tercera generación e implementan nuevas tecnologías gráficas para el “gamming” y el diseño gráfico, funcionando con un menor consumo y teniendo un mejor rendimiento a un buen precio. Continua como su predecesor en 22 nanómetros, pero funciona con un nuevo socket con clave 1150. Tienen un costo elevado a comparación con los APU's y FX de AMD pero tienen un mayor rendimiento.

2020: Intel Core S 10ª. Generación

Estos procesadores de décima generación se encuentran orientados para ordenadores 'gaming' de sobremesa, alcanzando una frecuencia de procesamiento máxima de 5,3 GHz en su modelo tope de gama, el i9-10900K.

2021: Intel Core de 11ª Generación

Estos procesadores cuentan con los nuevos transistores SuperFin, combinan nuevas tecnologías como WiFi 6, Thunderbolt 4, decodificación de medios AV1, interfaz PCI Express Gen 4 anexada al procesador y características de seguridad reforzadas por hardware. Admite velocidades de hasta 4.8 Ghz e Intel Optane H10 con almacenamiento en estado sólido para las unidades más veloces. Además, ofrecen aceleración de inteligencia artificial con su motor Intel Deep Learning Boost y gráficos Intel Iris X de calidad de tarjeta dedicada con miles de millones de colores, HDR 10, sonido Dolby Atmos y Dolby Vision con aceleración por hardware.[17]

2.12 MICROCONTROLADOR

Un microcontrolador (abreviado µC, UC o mCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales que cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida.

Algunos microcontroladores pueden utilizar palabras de cuatro bits y funcionan a velocidad de reloj con frecuencias tan bajas como 4 kHz, con un consumo de baja potencia (mW o microwatts). Por lo general, tendrá la capacidad de mantenerse a la espera de un evento como pulsar un botón o de otra interrupción; así, el consumo de energía durante el estado de reposo (reloj de la CPU y los periféricos de la mayoría) puede ser solo de nanowatts, lo que hace que muchos de ellos sean muy adecuados para aplicaciones con batería de larga duración. Otros microcontroladores pueden servir para roles de rendimiento crítico, donde sea necesario actuar más como un procesador digital de señal (DSP), con velocidades de reloj y consumo de energía más altos.

Cuando es fabricado el microcontrolador, no contiene datos en la memoria ROM. Para que pueda controlar algún proceso es necesario generar o crear y luego grabar en la EEPROM o equivalente del microcontrolador algún programa, el cual puede ser escrito

[17] Wikipedia, la enciclopedia libre, (2021). *Microprocesador*. Recuperado de https://es.wikipedia.org/wiki/Microprocesador

en lenguaje ensamblador u otro lenguaje para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser codificado en sistema numérico hexadecimal que es finalmente el sistema que hace trabajar al microcontrolador cuando este es alimentado con el voltaje adecuado y asociado a dispositivos analógicos y discretos para su funcionamiento.

2.12.1 HISTORIA

El primer microprocesador fue el Intel 4004 de 4 bits, lanzado en 1971, seguido por el Intel 8008 y otros más capaces. Sin embargo, ambos procesadores requieren circuitos adicionales para implementar un sistema de trabajo, elevando el costo del sistema total.

El Instituto Smithsoniano dice que los ingenieros de Texas Instruments Gary Boone y Michael Cochran lograron crear el primer microcontrolador, TMS 1000, en 1971; fue comercializado en 1974. Combina memoria ROM, memoria RAM, microprocesador y reloj en un chip y estaba destinada a los sistemas embebidos.2

Debido en parte a la existencia del TMS 1000, Intel desarrolló un sistema de ordenador en un chip optimizado para aplicaciones de control, el Intel 8048, que comenzó a comercializarse en 1977. combina memoria RAM y ROM en el mismo chip y puede encontrarse en más de mil millones de teclados de compatible IBM PC, y otras numerosas aplicaciones. El en ese momento presidente de Intel, Luke J. Valenter, declaró que el microcontrolador es uno de los productos más exitosos en la historia de la compañía, y amplió el presupuesto de la división en más del 25%.

La mayoría de los microcontroladores en aquel momento tenían dos variantes. Unos tenían una memoria EPROM reprogramable, significativamente más caros que la variante PROM que era solo una vez programable. Para borrar la EPROM necesita exponer a la luz ultravioleta la tapa de cuarzo transparente. Los chips con todo opaco representaban un coste menor.

En 1993, el lanzamiento de la EEPROM en los microcontroladores (comenzando con el Microchip PIC16x84) permite borrarla eléctrica y rápidamente sin necesidad de un paquete costoso como se requiere en EPROM, lo que permite tanto la creación rápida de

prototipos y la programación en el sistema. El mismo año, Atmel lanza el primer microcontrolador que utiliza memoria flash. Otras compañías rápidamente siguieron el ejemplo, con los dos tipos de memoria.

El costo se ha desplomado en el tiempo, con el más barato microcontrolador de 8 bits disponible por menos de 0,25 dólares para miles de unidades en 2009, y algunos microcontroladores de 32 bits a 1 dólar por cantidades similares. En la actualidad los microcontroladores son baratos y fácilmente disponibles para los aficionados, con grandes comunidades en línea para ciertos procesadores.

En el futuro, la MRAM podría ser utilizada en microcontroladores, ya que tiene resistencia infinita y el coste de su oblea semiconductora es relativamente bajo.

2.12.2 CARACTERÍSTICAS

Los microcontroladores están diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo como una batidora utilizará un procesador muy pequeño (4 u 8 bits) porque sustituirá a un autómata finito. En cambio, un reproductor de música o vídeo digital (MP3 o MP4) requerirá de un procesador de 32 bits o de 64 bits y de uno o más códecs de señal digital (audio o vídeo). El control de un sistema de frenos ABS (Antilock Brake System) se basa normalmente en un microcontrolador de 16 bits, al igual que el sistema de control electrónico del motor en un automóvil.

Los microcontroladores representan la inmensa mayoría de los chips de computadoras vendidos, sobre un 50% son controladores "simples" y el restante corresponde a DSP más especializados. Mientras se pueden tener uno o dos microprocesadores de propósito general en casa (Ud. está usando uno para esto), usted tiene distribuidos seguramente entre los electrodomésticos de su hogar una o dos docenas de microcontroladores. Pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

Un microcontrolador difiere de una unidad central de procesamiento normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de circuitos integrados externos de apoyo. La idea es que el circuito integrado se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips. Hay que agregarle los módulos de entrada y salida (puertos) y la memoria para almacenamiento de información.

Un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria de acceso aleatorio o ROM/EPROM/EEPROM/flash, con lo que para hacerlo funcionar todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidor analógico digital, temporizadores, UARTs y buses de interfaz serie especializados, como I2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos microcontroladores frecuentemente incluyen un lenguaje de programación integrado, como el lenguaje de programación BASIC que se utiliza bastante con este propósito.

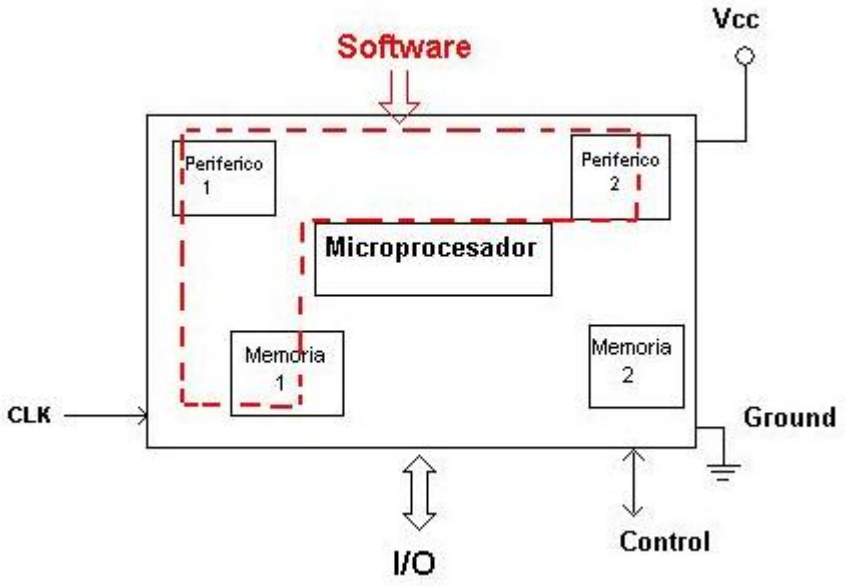


Figura 19: Esquema de un microcontrolador.

Fuente: https://commons.wikimedia.org/wiki/File:Microcontrolador.jpg

Los microcontroladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante sitio en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el microcontrolador, se ha de prescindir de cualquier otra circuitería.[18]

2.13 ARQUITECTURAS DE COMPUTADORAS

2.13.1 ARQUITECTURA DE VON NEUMANN

La arquitectura von Neumann, también conocida como modelo de von Neumann o arquitectura Princeton, es una arquitectura de computadoras basada en la descrita en 1945 por el matemático y físico John von Neumann y otros, en el primer borrador de un informe sobre el EDVAC. Este describe una arquitectura de diseño para un computador digital electrónico con partes que constan de una unidad de procesamiento que contiene una unidad aritmético lógica y registros del procesador, una unidad de control que contiene un registro de instrucciones y un contador de programa, una memoria para almacenar tanto datos como instrucciones, almacenamiento masivo externo, y mecanismos de entrada y salida. El concepto ha evolucionado para convertirse en un computador de programa almacenado en el cual no pueden darse simultáneamente una búsqueda de instrucciones y una operación de datos, ya que comparten un bus en común. Esto se conoce como el cuello de botella Von Neumann, y muchas veces limita el rendimiento del sistema.

El diseño de una arquitectura von Neumann es más simple que la arquitectura Harvard más moderna, que también es un sistema de programa almacenado, pero tiene un conjunto dedicado de direcciones y buses de datos para leer datos desde memoria y escribir datos en la misma, y otro conjunto de direcciones y buses de datos para ir a buscar instrucciones.

Un ordenador digital de programa almacenado es aquel que mantiene sus instrucciones de programa, así como sus datos, en una memoria de acceso aleatorio (RAM) de lectura-escritura. Las computadoras de programa almacenado representaron un avance sobre los

[18] Wikipedia, la enciclopedia libre, (2021). *Microcontrolador*. Recuperado de https://es.wikipedia.org/wiki/Microcontrolador

ordenadores controlados por programas de la década de 1940, como la Colossus y la ENIAC, que se programaron mediante el establecimiento de conmutadores y la inserción de cables de interconexión para enrutar datos y para controlar señales entre varias unidades funcionales. En la gran mayoría de las computadoras modernas, se utiliza la misma memoria tanto para datos como para instrucciones de programa, y la distinción entre von Neumann vs. Harvard se aplica a la arquitectura de memoria caché, pero no a la memoria principal.

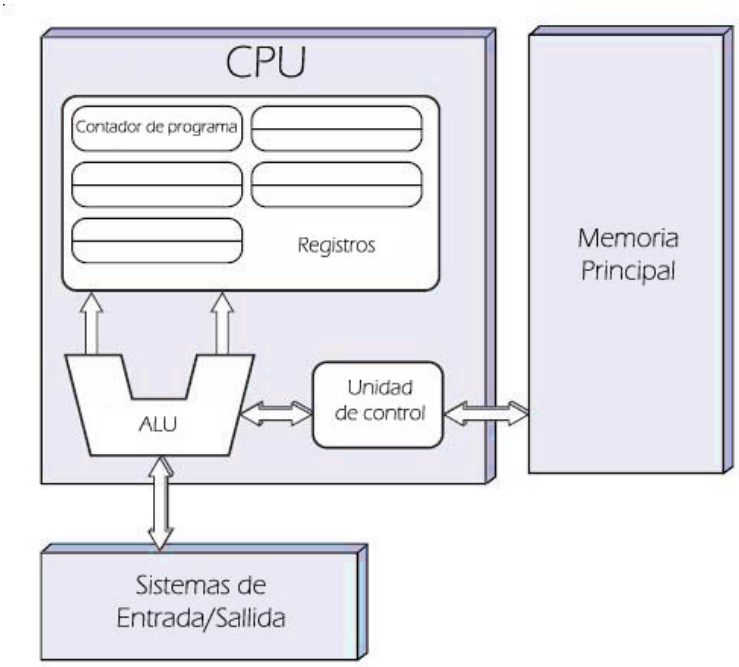


Figura 20: Arquitectura Von Neumann.
Fuente: https://commons.wikimedia.org/wiki/File:Arquitecturaneumann.jpg

2.13.1.1 HISTORIA

Las primeras máquinas de computación tenían programas fijos. Algunos equipos muy simples siguen utilizando este diseño, ya sea por motivos de simplificación o de formación. Por ejemplo, una calculadora de escritorio es (en principio) una computadora de programa fijo. En ella, se pueden hacer matemáticas básicas, pero no puede ser utilizada como procesador de texto o consola de juegos. Cambiar el programa de una máquina de programa fijo requiere re cablear, reestructurar, o re diseñar la máquina. Las primeras computadoras no eran tanto "programadas" ya que fueron "diseñadas"."Reprogramar" cuando era posible, era un proceso laborioso que comenzaba

con diagramas de flujo y notas de papel, seguido de diseños detallados de ingeniería y luego el muchas veces arduo proceso de recablear físicamente y reconstruir la máquina. Podía tomar hasta tres semanas preparar un programa de ENIAC y conseguir que funcionara.

Esa situación cambió con la propuesta de la computadora con programa almacenado. Una computadora de programa almacenado incluye, por diseño, un conjunto de instrucciones y puede almacenar en la memoria un conjunto de instrucciones (un programa) que detalla la computación.

Un diseño de programa almacenado también permite un código mutante. Una primera motivación para una instalación de este tipo fue la necesidad de que un programa incremente o modifique de otro modo la porción de dirección de instrucciones, lo cual, en los primeros diseños, tenía que hacerse manualmente. Esto llegó a ser menos importante cuando los registros índice y modos de direccionamiento se convirtieron en características habituales de la arquitectura de la máquina. Otro uso fue para incrustar datos frecuentemente usados en el flujo de instrucciones utilizando direccionamiento inmediato. El código mutante ha caído en gran parte en desuso, ya que suele ser difícil de entender y de depurar, además de ser ineficiente, en favor de los regímenes de los modernos procesadores pipelines y del almacenamiento en caché.

A gran escala, la capacidad para tratar a las instrucciones de la misma forma que si fueran datos es lo que hacen los ensambladores, compiladores, enlazadores, cargadores, y otras posibles herramientas automáticas de programación. Se puede "escribir programas que escriban programas". En una escala menor, las intensivas operaciones repetitivas de E/S –como los primeros manipuladores de imágenes BitBLT o los sombreadores de píxeles y vértices en los gráficos 3D modernos–, se consideraron ineficaces al funcionar sin necesidad de hardware personalizado. Estas operaciones podrían acelerarse en los procesadores de propósito general con tecnología de "compilación mosca" ("compilación en tiempo de ejecución"), por ejemplo, programas de código generado, una forma de código automodificable que ha mantenido popularidad.

Hay algunas desventajas para el diseño de von Neumann. Aparte del cuello de botella de von Neumann descrito a continuación, las modificaciones del programa pueden ser muy perjudiciales, ya sea por accidente o por diseño. En algunos diseños simples de computadora con programa almacenado, un programa que no funcione correctamente puede dañarse, dañar a otros programas, o inclusive al sistema operativo, lo que puede dar lugar a un desplome o crash de la computadora. Normalmente, la protección de memoria y otras formas de control de acceso pueden proteger tanto de modificaciones accidentales como de programas maliciosos.

2.13.1.2 DEFINICIÓN FORMAL

Las computadoras son máquinas de arquitectura von Neumann cuando:

1. Tanto los programas como los datos se almacenan en una memoria en común. Esto hace posible la ejecución de comandos de la misma forma que los datos.
2. Cada celda de memoria de la máquina se identifica con un número único, llamado dirección.
3. Las diferentes partes de la información (los comandos y los datos) tienen diferentes modos de uso, pero la estructura no se representa en memoria de manera codificada.
4. Cada programa se ejecuta de forma secuencial que, en el caso de que no haya instrucciones especiales, comienza con la primera instrucción. Para cambiar esta secuencia se utiliza el comando de control de transferencia.

2.13.1.3 EVOLUCIÓN

A lo largo de las décadas de los años 1960 y 1970, las computadoras se hicieron, en general, tanto más pequeñas como rápidas, lo que llevó a algunas evoluciones en su arquitectura. Por ejemplo, el mapeado en memoria de E/S permitió que los dispositivos de entrada y salida fueran tratados de la misma como la memoria. un único bus de sistema podría ser utilizado para proporcionar un sistema modular con un menor coste. A veces esto se denomina "racionalización" de la arquitectura. En las décadas siguientes, los microcontroladores sencillos permitirían algunas veces omitir características del

modelo a menor costo y tamaño. Las computadoras más grandes añadían características para un mayor rendimiento.[19]

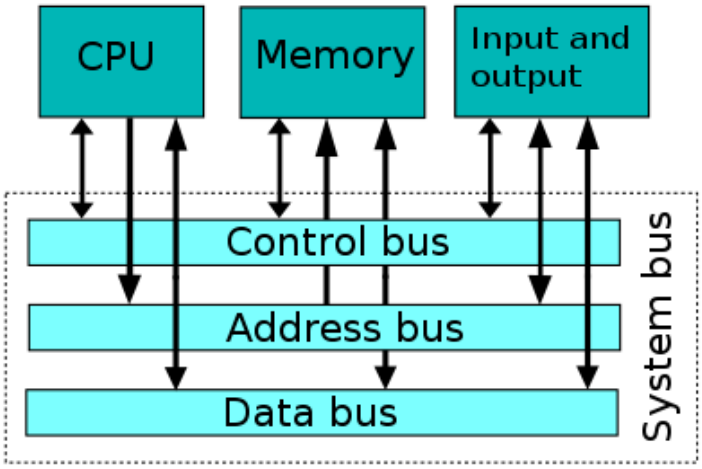


Figura 21: Bus Von Neumann.

Fuente: https://commons.wikimedia.org/wiki/File:Computer_system_bus.svg

2.13.2 ARQUITECTURA HARVARD

La arquitectura de Harvard es una arquitectura de computadora con pistas de almacenamiento y de señal físicamente separadas para las instrucciones y para los datos. El término proviene de la computadora Harvard Mark I basada en relés, que almacenaba las instrucciones sobre cintas perforadas (de 24 bits de ancho) y los datos en interruptores electromecánicos. Estas primeras máquinas tenían almacenamiento de datos totalmente contenido dentro la unidad central de proceso, y no proporcionaban acceso al almacenamiento de instrucciones como datos. Los programas necesitaban ser cargados por un operador; el procesador no podría arrancar por sí mismo.

En la actualidad la mayoría de los procesadores implementan dichas vías de señales separadas por motivos de rendimiento, pero en realidad implementan una arquitectura Harvard modificada, para que puedan soportar tareas tales como la carga de un programa desde una unidad de disco como datos para su posterior ejecución.

[19] Wikipedia, la enciclopedia libre, (2021). *Arquitectura de Von Neumann.* Recuperado de https://es.wikipedia.org/wiki/Arquitectura_de_Von_Neumann

En la arquitectura Harvard, no hay necesidad de hacer que las dos memorias compartan características. En particular, pueden diferir la anchura de palabra, el momento, la tecnología de implementación y la estructura de dirección de memoria. En algunos sistemas, se pueden almacenar instrucciones en memoria de solo lectura mientras que, en general, la memoria de datos requiere memoria de lectura-escritura. En algunos sistemas, hay mucha más memoria de instrucciones que memoria de datos así que las direcciones de instrucción son más anchas que las direcciones de datos.

2.13.2.1 CONTRASTE CON LA ARQUITECTURA VON NEUMANN

Bajo arquitectura de von Neumann pura, la CPU puede estar bien leyendo una instrucción o leyendo/escribiendo datos desde/hacia la memoria, pero ambos procesos no pueden ocurrir al mismo tiempo, ya que las instrucciones y datos usan el mismo sistema de buses. En una computadora que utiliza la arquitectura Harvard, la CPU puede tanto leer una instrucción como realizar un acceso a la memoria de datos al mismo tiempo, incluso sin una memoria caché. En consecuencia, una arquitectura de computadores Harvard puede ser más rápida para un circuito complejo, debido a que la instrucción obtiene acceso a datos y no compite por una única vía de memoria.

Además, las características de las dos memorias son distintas, por lo que la dirección del espacio cero de instrucciones no es lo mismo que la dirección del espacio cero de datos: La dirección cero de la memoria de instrucciones podría identificar un valor de veinticuatro bits, mientras que la dirección cero de la memoria de datos cero podría indicar un valor de ocho bits que no forma parte de ese valor de veinticuatro bits.

2.13.2.2 CONTRASTE CON LA ARQUITECTURA HARVARD MODIFICADA

Una máquina de arquitectura Harvard modificada es muy similar a una máquina de arquitectura Harvard, pero relaja la estricta separación entre la instrucción y los datos, al mismo tiempo que deja que la CPU acceda simultáneamente a dos (o más) memorias de buses. La modificación más común incluye cachés de instrucciones y datos independientes, respaldados por un espacio de direcciones en común. Si bien la CPU

ejecuta desde la memoria caché, también actúa como una máquina de Harvard pura. Cuando se accede a la memoria de respaldo, actúa como una máquina de von Neumann pura (donde el código puede moverse alrededor como datos, que es una técnica poderosa). Esta modificación se ha generalizado en modernos procesadores, tales como la arquitectura ARM y los procesadores x86. A veces se llama vagamente arquitectura Harvard, con vistas al hecho de que en realidad está "modificada".

Otra modificación proporciona un camino entre la memoria de instrucciones (como ROM o flash) y la CPU para permitir que las palabras de la memoria de instrucciones sean tratados como datos de solo lectura. Esta técnica es utilizada en algunos micro controladores, incluyendo el Atmel AVR. Esto permite datos constantes, tales como cadenas de texto o tablas de funciones, que puede acceder sin necesidad de ser previamente copiadas en datos de memoria, preservando memoria de datos escasa (y hambrienta de poder) de lectura / escritura de variables. Las instrucciones especiales de lenguaje de máquina se proporcionan para leer datos desde la memoria de instrucciones. (Esto es diferente a las instrucciones que a sí mismos embebiendo datos constantes, aunque para las constantes individuales de los dos mecanismos pueden sustituir unos por otros.)[20]

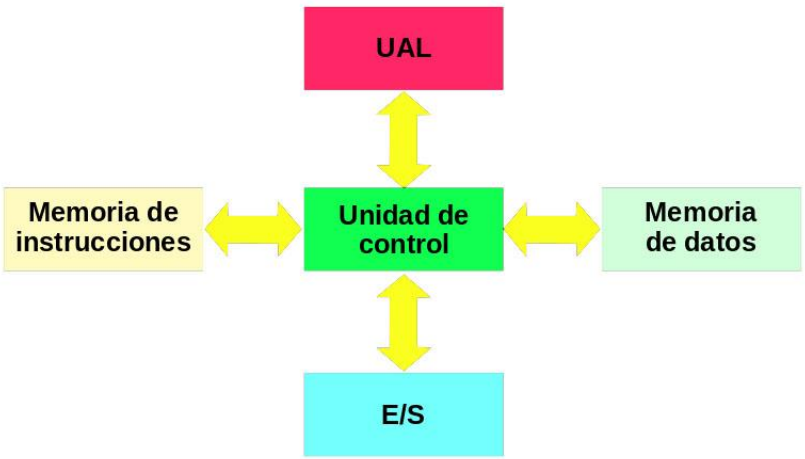


Figura 22: Arquitectura Harvard.

Fuente: https://commons.wikimedia.org/wiki/File:Harvard_architecture-es.svg

[20] Wikipedia, la enciclopedia libre, (2021). *Arquitectura Harvard*. Recuperado de https://es.wikipedia.org/wiki/Arquitectura_Harvard

2.14 INTERFACES

En electrónica, telecomunicaciones y hardware, una interfaz es el puerto (circuito físico) a través del que se envían o reciben señales desde un sistema o subsistemas hacia otros. No existe una interfaz universal, sino que existen diferentes estándares (Interfaz USB, interfaz SCSI, etc.) que establecen especificaciones técnicas concretas (características comunes), con lo que la interconexión sólo es posible utilizando la misma interfaz en origen y destino. Así también, una interfaz puede ser definida como un intérprete de condiciones externas al sistema, a través de transductores y otros dispositivos, que permite una comunicación con actores externos, como personas u otros sistemas, a través de un protocolo común a ambos. Una interfaz es una Conexión física y funcional entre dos aparatos o sistemas independientes.

2.14.1 OBJETIVO DE LOS DISPOSITIVOS E/S

La computación de entrada salida, o E/S, se refiere a la comunicación entre un sistema de procesamiento de información (como un computador), y los agentes humanos u otro sistema de procesamiento de información. Las entradas son las señales o datos recibidos por el sistema, y salidas son las señales enviadas por este.

Un dispositivo de E/S es un componente electrónico que permite la transmisión y/o recepción de información de/hacia el ordenador. Como ejemplo el ratón y el teclado son dispositivos de entrada, y el monitor y la impresora son dispositivos de salida. Los dispositivos para comunicación entre computadores son típicamente dispositivos de entrada y de salida.

En la arquitectura de computadores la combinación entre la CPU y la memoria principal está considerada el cerebro de la computadora y desde este punto de vista cualquier transferencia de información desde el computador es considerada entrada, y hacia el computador es considerada Salida.

El objetivo principal es interconectar la mayor cantidad de dispositivos a un computador, pero hay que atender a las distintas características que presentan cada uno de ellos y que a menudo suelen diferir de las propias del procesador, podemos destacar:

- Tienen, normalmente, menor velocidad que el procesador.
- La longitud de palabra.
- Los códigos que cada uno de ellos emplean para la representación de datos.

La interfaz de E/S es requerida cuando los dispositivos son ejecutados por el procesador. La interfaz debe ser necesariamente lógica para interpretar la dirección de los dispositivos generados por el procesador. El Handshaking deberá ser implementado por la interfaz usando los comandos adecuados (BUSY, READY, WAIT…), y el procesador puede comunicarse con el dispositivo de E/S a través de la interfaz. Si se intercambian diferentes formatos de datos, la interfaz debe ser capaz de convertir datos en serie a paralelo y viceversa. Los dispositivos de E/S se comunican por interrupciones con el procesador, si una interrupción es recibida, el procesador la atenderá con la rutina de interrupción correspondiente a dicha interrupción.

Un ordenador que usa E/S mapeados en memoria por lectura y escritura accede al hardware a través de la posición de memoria específica, usando el mismo lenguaje ensamblador que el procesador usa para el acceso a memoria.

2.14.2 CONTROLADOR DE PERIFÉRICO

Actualmente se usan multitud de interfaces o controladores para las conexiones entre el procesador y los distintos periféricos (cada uno de estos últimos suele tener su propio controlador). En ocasiones se puede interconectar los periféricos con la memoria principal directamente sin pasar por el procesador para lo cual se utilizan dispositivos más avanzados como los DMA que son procesadores dedicados a dichas transferencias.

Estos dispositivos tratan de permitir la transferencia de datos hacia/desde el periférico determinado. Entre sus principales características podemos destacar:

- Tienen diversos puertos asociados. Entendemos por puerto algo que puede ser referenciado y accedido a través de una dirección (no tiene por qué ser un hardware específico de almacenamiento, aunque en la mayoría de los casos suelen ser registros).
- Poseen circuitería específica para la adaptación del formato de señales y de velocidades entre el procesador y los dispositivos de E/S.
- Proporcionan las transferencias de datos, como especificamos anteriormente, usando el bus de datos.
- Requieren programas software para el proceso de transferencia, que será ejecutado por el procesador cada vez que se requiera usar al periférico involucrado.
- En computadores de alta gama se pueden emplear controladores más sofisticados que son en realidad procesadores específicos que solo tienen funciones para la E/S, son los llamados canales o IOP.

2.14.3 FUNCIONES BÁSICAS DE UN SISTEMA E/S

Un sistema de E/S debe poder, como mínimo, direccionar los diferentes periféricos con los que puede establecer comunicación, establecer un sistema de comunicación entre el procesador y los controladores, y sincronizar los dispositivos de manera que no se produzcan inconsistencias o errores. Además, debería ser capaz de convertir los datos entre diferentes formatos, controlar el estado de los periféricos, llevar la cuenta de las transmisiones y tener un sistema de detección de errores.

2.14.3.1 DIRECCIONAMIENTO O SELECCIÓN DE PERIFÉRICO

A la hora de comunicarse con un dispositivo, el procesador sitúa su dirección asociada en el bus de direcciones. De esta forma, selecciona el dispositivo con el que quiere iniciar una transferencia de datos. La forma de identificar cada periférico es el llamado direccionamiento.

Tiene las siguientes características:

- Es necesario prevenir que varios dispositivos puedan acceder simultáneamente al mismo bus, ya que se pueden producir cortocircuitos (en el caso de una arquitectura de bus compartido).
- Cada dirección no tiene por qué referirse a un solo puerto, pero es necesario que ésta se pueda identificar unívocamente y no produzca incoherencias o ambigüedades con el resto del sistema.
- Los controladores suelen tener varios puertos asociados.

Bus de direcciones

El bus de direcciones comunica el procesador con los periféricos, seleccionando aquel con el que se desea interaccionar. Este puede ser:

- Mapeado: El mapa de periféricos pertenece al mismo que el de memoria, es decir, el procesador no distingue entre accesos a memoria y accesos a los dispositivos de E/S, por tanto, no hay instrucciones específicas de E/S que no sean las propias de acceso a memoria, sino que se utilizan LOAD, STORE o MOVE. La ventaja de usar una E/S mapeada es que hay menor complejidad a la hora de diseñar el procesador.
- Independiente: El mapa de periféricos es independiente (y valga la redundancia) al mapa de memoria ya que usa la plantilla IO/M# del procesador, por tanto existen instrucciones dedicadas a las transferencias con periféricos distintos a la memoria principal (nemotécnicos más comunes: IN, OUT, TESTI/O, CONTROLI/O). Las ventajas de usar E/S independiente es que, facilita la protección de E/S y los programas son más rápidos al tener una decodificación más sencilla y tener un tamaño menor las instrucciones de E/S. Al contrario que la E/S mapeada, el diseño del procesador es más complejo.

Técnicas de direccionamiento para controladores

En función del tipo de sistema E/S, podemos encontrar diferentes formas de direccionamiento:

- Direccionamiento por selección lineal: Consiste en asignar un bit del bus de direcciones a cada puerto. Si tenemos n "líneas" para direccionar periféricos podemos especificar, a lo sumo, n periféricos distintos. Este tipo de direccionamiento no se suele utilizar ya que tiene importantes restricciones tales como que solo un periférico puede estar activo a la vez.
- Direccionamiento por selección por decodificación: La dirección del periférico está codificada y se requiere un decodificador para activar el periférico deseado. Dentro de este tipo de direccionamiento se hallan dos posibilidades:
    - Centralizado: Se emplea un decodificador para todos los puertos.
    - Distribuido: Cada puerto "reconoce" su propia dirección.

2.14.3.2 COMUNICACIÓN FÍSICA ENTRE EL CONTROLADOR Y EL PROCESADOR

Existen distintas formas de interconexión que se pueden dar entre controlador y procesador. Las más destacadas son las basadas en buffer tri-estado y en las MUX/DEMUX indicando sus ventajas/inconvenientes.

Se suele usar más la alternativa basada en buffer tri-estado pues permite un mejor aprovechamiento de los dispositivos de E/S y la mejora de los mismos al dedicar mayor área de estos en la mejora de prestaciones y no en el interconexionado.

Basados en buffer tri-estado

Se implementan usando un bus compartido y buffer tri-estado para cada puerto y evitar así el "volcado" de información por parte de dos o más periféricos en el bus. Las características principales de este tipo de interconexión son:

- Facilidad en la expansión por medio de tarjetas o circuitería específica
- Permite conectar en paralelo muchos periféricos

Basados en MUX/DEMUX

Se emplean MUX y DEMUX para seleccionar el periférico que podrá usar el bus compartido en un momento dado, impidiendo al resto de dispositivos acceder a este último. Las características principales de este tipo de interconexión son:

- Escasa posibilidad de expansión
- Mucha circuitería: Suelen dedicar gran parte del área del dispositivo en el cableado del mismo.

2.14.3.3 SINCRONIZACIÓN

La sincronización con el procesador o la memoria de los dispositivos de E/S consiste en la acomodación de velocidades de ambos, ya que los periféricos suelen ser más lentos. A fin de que no se imponga el ritmo del dispositivo más lento se establecen mecanismos para saber cuándo se deben enviar/recibir datos y es común que los controladores dispongan de buffers de almacenamiento temporal de palabras y permitan aceptar señales de control de conformidad que reflejan el estado del periférico en un momento dado (listo, petición, reconocimiento, ocupado, etc).

Tipos de temporización

La temporización de las operaciones de E/S puede ser de dos tipos:

- Síncrona: Aquella en la que los dispositivos que se conectan poseen velocidades similares, por lo que ninguno de ellos debe esperar al otro.
- Asíncrona: Aquella en los que los dispositivos tienen velocidades dispares, lo cual provoca que uno deba esperar al otro para que no se produzcan errores/inconsistencias en los datos.

Existe otra definición de los términos Sincrono/Asincrono que se puede encontrar en diferentes textos/escritos de diferentes autores dichos términos refiriéndose a las distintas definiciones:

- Síncronos: Se dice de los dispositivos que comparten una señal de reloj común.

- Asíncronos: Aquellos que no comparten la misma señal de reloj. Debido a que no tienen la misma señal de reloj, pueden acomodar una amplia variedad de dispositivos, y el bus puede alargarse sin preocuparse por los problemas de sincronización. Para coordinar la transmisión de datos entre emisor y receptor, utilizan un protocolo de presentación (handshaking protocol). Éste protocolo consta de una serie de pasos en los cuales emisor y receptor proceden al siguiente paso solamente cuando ambas partes están de acuerdo. Para implementar éste protocolo se requiere de un conjunto adicional de líneas de control, entre ellas:
  1. ReadReq o RD: Se utiliza para indicar una petición de lectura de memoria. La dirección se pone en la línea de datos al mismo tiempo.
  2. WriteReq o WR: Se utiliza para indicar una petición de escritura en memoria.
  3. Data: Se utiliza para indicar que la palabra de datos está preparada en las líneas de datos (datos estables).
  4. Ack: Se utiliza para conocer la señal de ReadReq o Data, es decir, esta señal de aceptación se necesita para que el controlador del periférico conteste a la petición de transferencia generada por el procesador.

Temporización síncrona

En la temporización síncrona, la aparición de un evento está determinada por el reloj. El bus incluye una línea de reloj que es común a todos los dispositivos, y se suelen sincronizar durante el flanco de subida. Casi todos los eventos duran un único ciclo de reloj.

Este tipo de temporización sólo funciona si el dispositivo de E/S es suficientemente rápido para responder a la espera que le brinda el procesador, en caso contrario se producirán errores en la escritura de los puertos y se leerán datos no válidos del bus, por lo que las operaciones de E/S no serían correctas. Otro tema importante es que, si no se incorporan rutinas para detectar hardware inexistente, el programa podría fallar por tratar de direccionar un dispositivo que o bien no existe o bien ha sido "desconectado" del equipo.

Las operaciones de lectura y escritura funcionan de la siguiente manera:

- Escritura: El procesador activa la señal WR#, que es la de escritura, y espera un tiempo, que es determinado por el procesador (no es una espera al otro dispositivo), y una vez transcurrido ese periodo de tiempo desactiva la señal de escritura, por lo que si el dispositivo de E/S no es suficientemente rápido la escritura no se realizará correctamente. Cabe destacar que en este tipo de temporización no se produce ninguna espera por parte del procesador ni por parte del periférico.
- Lectura: El procesador activa la señal RD# que es la de lectura y espera un tiempo determinado por el mismo (como ocurría en el caso de la Escritura) y acto seguido lee del bus de datos la información, sin comprobar si esta contiene los datos válidos suministrados por el periférico.

Temporización asíncrona o con "handshaking"

La traducción de "handshaking" es "apretón de manos" y viene a significar que el procesador y los periféricos intercambian señales de control que les permiten sincronizar sus acciones y "colaborar" conjuntamente en la transferencia de información. Generalmente se suele considerar que existe sólo una señal de sincronización llamada ACK (aunque puede haber tantas señales de sincronización como se necesiten y esto depende del hardware del dispositivo en cuestión).

Con este procedimiento mejoramos el rendimiento de las operaciones de E/S e impedimos que se produzcan los fallos en la escritura/lectura que podían suceder con la temporización síncrona para un dispositivo existente. Sin embargo, al igual que en el caso sincrono se han de incorporar rutinas que determinen si se intenta acceder a un dispositivo inexistente o "desconectado" puesto que esto si provocaría errores (entraría en un bucle infinito esperando a ACK). La solución que generalmente se adopta es que el procesador da un tiempo límite de espera por encima del cual se genera una excepción y se aborta la operación E/S.

El funcionamiento de las operaciones de lectura y escritura es el siguiente:

- Escritura: El procesador activa la señal de escritura, WR# y espera hasta que el periférico activa su línea de sincronización ACK, cuando esto sucede el

procesador deshabilita la señal de escritura y se produce la escritura (y valga la redundancia) de los datos en el puerto del periférico correspondiente. Finalmente, el periférico desactiva ACK.

- Lectura: El procesador habilita la señal de lectura, RD# y espera hasta que el dispositivo le "indique" mediante ACK que los datos están listos para su lectura. Una vez que ACK está activa el procesador lee los datos del bus de datos y desactiva la señal RD#. Finalmente, el periférico reconoce la deshabilitación de RD# y este desactiva la señal de sincronización, ACK.[21]

2.15 LENGUAJE ENSAMBLADOR

El lenguaje ensamblador o assembly (en inglés: assembly language y la abreviación asm) es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste

[21] Wikipedia, la enciclopedia libre, (2021). *Interfaz (Electrónica).* Recuperado de https://es.wikipedia.org/wiki/Interfaz_(electr%C3%B3nica)

con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macro ensambladores.

Fue usado principalmente en los inicios del desarrollo de software, cuando aún no se contaba con potentes lenguajes de alto nivel y los recursos eran limitados. Actualmente se utiliza con frecuencia en ambientes académicos y de investigación, especialmente cuando se requiere la manipulación directa de hardware, alto rendimiento, o un uso de recursos controlado y reducido. También es utilizado en el desarrollo de controladores de dispositivo (en inglés, device drivers) y en el desarrollo de sistemas operativos, debido a la necesidad del acceso directo a las instrucciones de la máquina. Muchos dispositivos programables (como los microcontroladores) aún cuentan con el ensamblador como la única manera de ser manipulados.

```
ORG    8030H
include
T05SEG:
   SETB TR0
   JNB uSEG,T05SEG ;esta subrutina es utilizada
   CLR TR0 ;para realizar una cuenta de
   CPL uSEG ;0,5 segundos mediante la
   MOV R1,DPL ;interrupción del timer 0.
       INVOKE
   MOV R2,DPH
   CJNE R2,#07H,T05SEG
   CJNE R1,#78H,T05SEG
   MOV DPTR,#0
RET
```

Figura 23: Código en lenguaje ensamblador para µC Intel 80C51.
Fuente: https://es.wikipedia.org/wiki/Lenguaje_ensamblador

2.15.1 CARACTERÍSTICAS

El código escrito en lenguaje ensamblador posee una cierta dificultad de ser entendido ya que su estructura se acerca al lenguaje máquina, es decir, es un lenguaje de bajo nivel.

El lenguaje ensamblador es difícilmente portable, es decir, un código escrito para un microprocesador, puede necesitar ser modificado, para poder ser usado en otra máquina distinta. Al cambiar a una máquina con arquitectura diferente, generalmente es necesario reescribirlo completamente.

Los programas hechos por un programador experto en lenguaje ensamblador pueden ser más rápidos y consumir menos recursos del sistema (ej: memoria RAM) que el programa equivalente compilado desde un lenguaje de alto nivel. Al programar cuidadosamente en lenguaje ensamblador se pueden crear programas que se ejecutan más rápidamente y ocupan menos espacio que con lenguajes de alto nivel. Conforme han evolucionado tanto los procesadores como los compiladores de lenguajes de alto nivel, esta característica del lenguaje ensamblador se ha vuelto cada vez menos significativa. Es decir, un compilador moderno de lenguaje de alto nivel puede generar código casi tan eficiente como su equivalente en lenguaje ensamblador.

Con el lenguaje ensamblador se tiene un control muy preciso de las tareas realizadas por un microprocesador por lo que se pueden crear segmentos de código difíciles y/o muy ineficientes de programar en un lenguaje de alto nivel, ya que, entre otras cosas, en el lenguaje ensamblador se dispone de instrucciones del CPU que generalmente no están disponibles en los lenguajes de alto nivel.[22]

2.16 LENGUAJE C

C es un lenguaje de programación de propósito general2:1 originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell,1 como evolución del anterior lenguaje B, a su vez basado en BCPL.2:134.

Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear softwares de sistemas y aplicaciones.

[22] Wikipedia, la enciclopedia libre, (2021). *Lenguaje Ensamblador*. Recuperado de https://es.wikipedia.org/wiki/Lenguaje_ensamblador

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipado, de medio nivel, que dispone de las estructuras típicas de los lenguajes de alto nivel, pero, a su vez, dispone de construcciones del lenguaje que permiten un control a bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas.

2.17 COMPILADOR C PARA PEQUEÑOS DISPOSITIVOS SDCC

El compilador SDCC (por sus siglas en inglés Small Device C Compiler (SDCC)) es un compilador reorientable de Software libre para lenguaje C enfocado en microcontroladores de 8 bits. Se distribuye bajo la Licencia Pública General GNU. El paquete también contiene un ensamblador, enlazador, simulador y depurador. En marzo de 2007, SDCC se volvió el único compilador de lenguaje C de código abierto para microcontroladores Intel 8051 y compatibles.123 En el año 2011 el compilador tuvo una tasa diaria promedio de descargas superior a 200.

Los archivos binarios, documentación y recursos se encuentran disponibles para sistemas Linux (32-bit y 64-bit), macOS (PPC y 64-bit) así como Windows (32-bit y 64-bit).

2.17.1 DISPOSITIVOS SOPORTADOS

SDCC tiene soporte para un gran número de dispositivos de diferentes fabricantes, entre los cuales se menciona los más relevantes:

- Intel 8031, 8032, 8051, 8052.
- Maxim/Dallas DS80C390.
- Motorola/Freescale/NXP 68HC08 y 68HCS08.

- Padauk PDK14 y PDK15.4.
- STMicroelectronics STM8.
- Zilog Z80, Z180, eZ80 bajo el set de comandos del microcontrolador Z80.
- Rabbit Semiconductor 2000, 2000A, 3000, 3000A, 4000.
- Sharp LR35902 (Procesador usado por el Game Boy).
- Toshiba TLCS-90.
- Z80N (ZX Spectrum Next processor).
- Microchip PIC16, PIC18.

2.18 BOOTLOADER

Un bootloader o gestor de arranque es un software especial que carga en la memoria interna el sistema operativo instalado en el sistema. Para conseguirlo, el bootloader suele ejecutarse directamente al arrancar un dispositivo usando algún medio que sea booteable, es decir, que sirva como unidad de arranque, como puede ser un disco duro, un CD o DVD, o un stick USB. El medio de arranque recibe la información acerca de dónde se encuentra el bootloader por parte del firmwaredel ordenador (BIOS, por ejemplo). Todo el proceso es lo que se denomina inicio, arranque o, en inglés booten.

En el caso especial de los microcontroladores, el bootloader se carga por lo general al final de la memoria de programa y este es capaz de gestionar otros programas que serán cargados al microcontrolador, usando vectores que apuntan a la dirección donde se encuentra el programa que se desea ejecutar.

2.19 VISUAL STUDIO COMMUNITY EDITION

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.[23]

Figura 24: Icono de Visual Studio.NET.
Fuente: https://commons.wikimedia.org/wiki/File:Visual_Studio_Icon_2019.svg

2.20 VISUAL BASIC.NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre el framework .NET. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es retro compatible con Visual Basic, pero el manejo de las instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas. Para mantener eficacia en el desarrollo de las aplicaciones. La gran mayoría de programadores de VB.NET utilizan el entorno de desarrollo integrado Microsoft Visual Studio en alguna de sus versiones (desde el primer Visual Studio .NET hasta Visual Studio .NET 2019, que es la última versión de Visual Studio para la plataforma .NET), aunque existen otras alternativas, como SharpDevelop (que además es libre). Al igual que con todos los lenguajes de programación basados en .NET, los programas escritos en VB .NET requieren el Framework .NET o Mono para ejecutarse.[24]

[23] Wikipedia, la enciclopedia libre, (2021). *Microsoft Visual Studio.* Recuperado de https://es.wikipedia.org/wiki/Microsoft_Visual_Studio
[24] Wikipedia, la enciclopedia libre, (2021). *Visual Basic.NET.* Recuperado de https://es.wikipedia.org/wiki/Visual_Basic_.NET

2.20.1 TIPOS DE DATOS

Todos los lenguajes de programación que cumplen las normas de .NET tienen muchas cosas en común, una de ellas es el conjunto de tipos de datos. Hay que destacar que estos tipos de datos están implementados como clases, de manera que una variable declarada de un tipo determinado, tendrá la capacidad de usar tanto los métodos como las propiedades que pertenezcan a la clase del tipo de dato.[25]

En la siguiente tabla se muestra una relación de los tipos de datos de .NET Framework y su correspondencia en VB.NET y C#.

Nombre de la clase	**Tipo de dato en VB.NET**	**Tipo de dato en C#**	**Descripción**
Byte	Byte	Byte	Entero sin signo de 8 bit.
Sbyte	Sbyte (No nativo)	sbyte	Entero sin signo de 8bit (Tipo no acorde con el CLS)
Int16	Short	short	Entero con signo de 16 bit.
Int32	Integer	int	Entero con signo de 32 bit.
Int64	Long	long	Entero con signo de 64 bit.
Uint16	UInt16 (No nativo)	ushort	Entero sin signo de 16 bit. (Tipo no acorde con el CLS)
Uint32	Uint32 (No nativo)	uint	Entero sin signo de 32 bit. (Tipo no acorde con el CLS)
Uint64	Uint64 (No nativo)	ulong	Entero sin signo de 64 bit. (Tipo no acorde con el CLS)
Single	Single	float	Numero con coma flotante de precisión simple, de 32 bit.
Double	Double	double	Numero con coma flotante de precisión doble, de 64 bit.
Boolean	Boolean	bool	Valor logico
Char	Char	char	Carácter unicode de 16 bit.
Decimal	Decimal	decimal	Valor decimal de 96 bit.
IntPtr	IntPtr (No nativo)	--	Entero con signo cuyo tamaño depende de la plataforma: 32 bit en plataformas de 32 bit y 64 bit en plataformas de 64 bit. (Tipo no acorde con el CLS)
UintPtr	UintPtr (No nativo)	--	Entero sin signo cuyo tamaño depende de la plataforma: 32 bit en plataformas de 32 bit y 64 bit en plataformas de 64 bit. (Tipo no acorde con el CLS)
String	String	string	Cadena de caracteres.

Tabla 8: Tipos de datos en .NET

Fuente: https://desarrolloweb.com/articulos/1388.php

[25] Desarrollo Web. (2021). *Tipos de datos en .NET*. Recuperado de https://desarrolloweb.com/articulos/1388.php

2.21 PROTEUS DESIGN SUITE

Proteus Design Suite es software de automatización de diseño electrónico, desarrollado por Labcenter Electronics Ltd., que consta de los dos programas principales: Ares e Isis, y los módulos VSM y Electra.

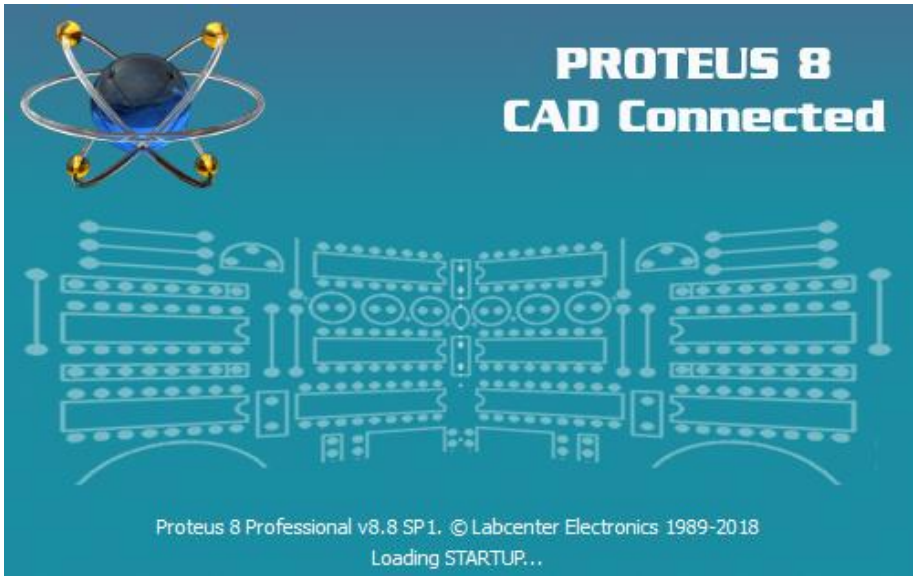


Figura 25: Presentación de inicio de Proteus 8.

Fuente: Captura de pantalla.

Proteus compone dos subprogramas los cuales son ISIS y ARES. Una de las prestaciones de Proteus, integrada con ISIS, es VSM, el Virtual System Modeling (Sistema Virtual de Modelado), una extensión integrada con ISIS, con la cual se puede simular, en tiempo real.

ARES, o Advanced Routing and Editing Software (Software de Edición y Ruteo Avanzado); es la herramienta de enrutado, ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso, permitiendo editar generalmente, las capas superficiales (Top Copper), y de soldadura (Bottom Copper).[26]

[26] Wikipedia, la enciclopedia libre, (2021). *Proteus Design Suite.* Recuperado de https://es.wikipedia.org/wiki/Proteus_Design_Suite

2.22 FAMILIA MCS-51

El Intel 8051 es un microcontrolador (µC) desarrollado por Intel en 1980 para uso en productos embebidos. Es un microcontrolador muy popular.

Los núcleos 8051 se usan en más de 100 microcontroladores de más de 20 fabricantes independientes como Atmel, Dallas Semiconductor, Philips, Winbond, entre otros.

La denominación oficial de Intel para familia de µCs 8051 es MCS 51.

Figura 26: INTEL 8051 en empaque DIP 40.
Fuente: https://commons.wikimedia.org/wiki/File:KL_Intel_P8051.jpg

2.22.1 ESPECIFICACIONES

2.22.1.1 HARDWARE

Este microcontrolador está basado en la Arquitectura de von Neumann con memoria segregada (es decir, existen espacios de direcciones separados para código y datos, pero las memorias comparten los buses internos de datos y direcciones). Aunque originariamente fue diseñado para aplicaciones simples, se permite direccionar 64 KB de ROM externa y 64 KB de RAM por medio de líneas separadas chip select para programa y datos.

Adicionalmente, el microcontrolador contiene una memoria interna, dividida en dos partes: los SFR y memoria de propósito general. Los SFR (Special Function Registers), son los registros proporcionados por el microcontrolador, y tienen asignadas direcciones en esta memoria interna. El acceso a esta memoria interna es más rápido que el acceso a

la memoria externa, pero es de tamaño limitado. Parte de esta memoria interna además se usa como pila durante las llamadas a función y el proceso de interrupciones.

Una característica particular del 8051 es la inclusión de una unidad de proceso booleano que permite que operaciones de nivel de bit lógica booleana se ejecuten directa y eficientemente en registros internos. Esto ha hecho que el 8051 sea muy popular en aplicaciones de control industrial.

Otra característica muy valorada es que tiene cuatro conjuntos separados de registros. A menudo se usa esta característica para reducir la latencia de interrupción. (La rutina que maneja la interrupción declara usar otro conjunto de registros, evitándose de esta manera tener que salvar en la pila los registros originales).

La mayoría de los 8051 incluyen una o dos UARTs, dos o tres temporizadores, 128 o 256 bytes de RAM interna (16 bytes de los cuales son direccionables a nivel de bit), cuatro o cinco registros de entrada/salida y entre 0k-54K de memoria interna de programa. El núcleo 8051 original ejecuta un ciclo máquina cada 12 ciclos de reloj, requiriendo la mayoría de instrucciones uno o dos ciclos máquina. Pero actualmente la mayoría de fabricantes ofrecen versiones mejoradas que solo requieren de 2 a 4 ciclos de reloj por cada instrucción máquina.

Los microcontroladores 8051 modernos ofrecen muchas mejoras sobre el original. Mejoras comunes incluyen watchdog timers (un temporizador programable que "resetea" el microcontrolador si no se refresca en cierto tiempo), osciladores internos, memoria de programa Flash ROM interna, código de inicialización en ROM, almacenamiento en EEPROM interna, I²C, SPI, USB, generadores PWM, conversores analógicos A/D y D/A, relojes de tiempo real RTC, temporizadores y contadores extra, facilidades de depuración internas, más fuentes de interrupción, modos de bajo consumo, interfaz CAN, etc.

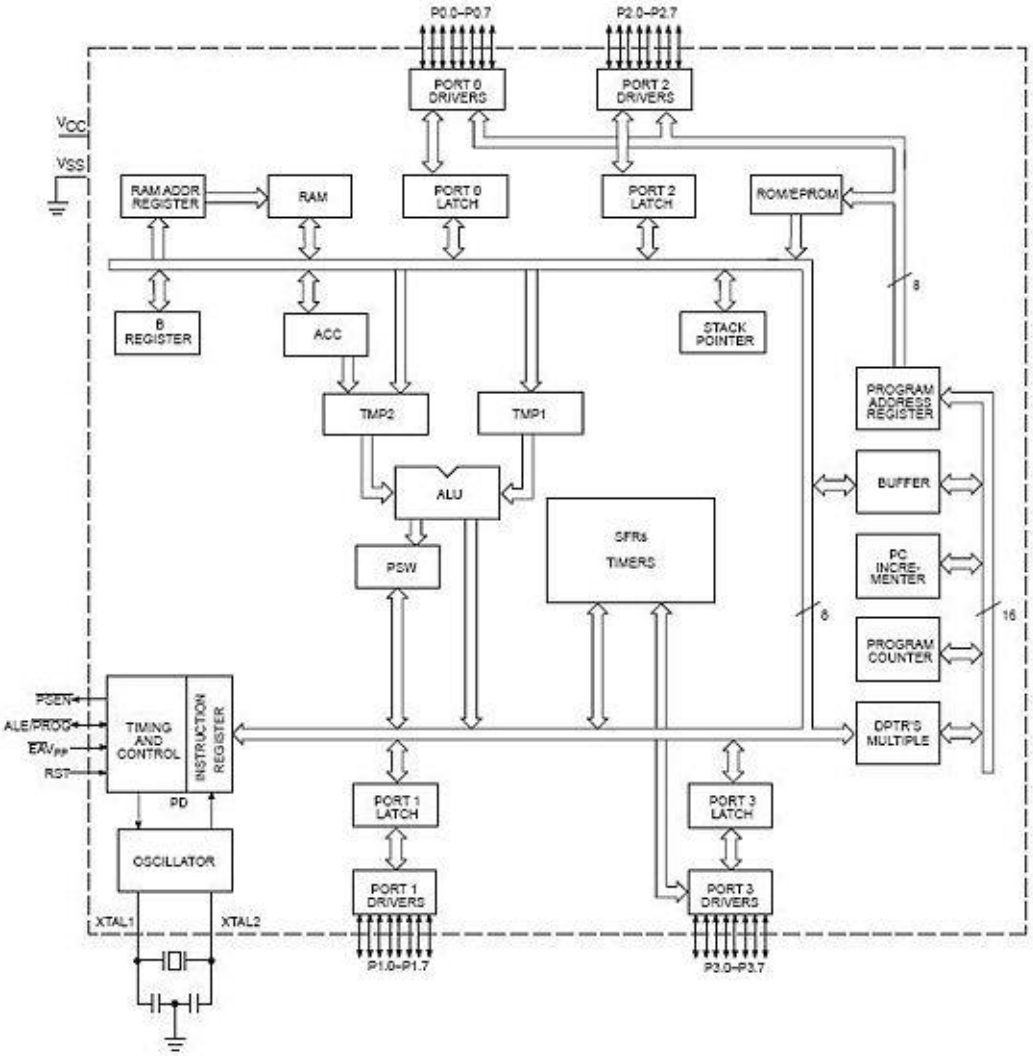


Figura 27: Diagrama de bloques 8051.

Fuente: https://commons.wikimedia.org/wiki/File:Diag_bloques_8051.JPG

2.22.1.2 SOFTWARE

Existen varios compiladores de lenguaje C disponibles para el 8051, así como en lenguaje ensamblador. La mayoría añaden extensiones al lenguaje que permiten al programador especificar por ejemplo el tipo de memoria donde se ubicará la variable, especificar manejadores de interrupción, especificar el banco de registros, acceder a instrucciones especiales de manipulación de bits...

El predecesor del 8051, el 8048, se usó en el teclado del primer IBM PC, donde convertía pulsaciones de tecla en un stream serie que se envía a la unidad central del ordenador. El 8048 y sus derivados aún se usan actualmente en teclados básicos.[27]

2.22.1.3 SET DE INSTRUCCIONES

Conjunto de Instrucciones en lenguaje ensamblador de la Familia de Microcontroladores 8051.[28]

ACALL Absolute Call / Llamada Absoluta.

ADD Add Accumulator / Añadir al Acumulador.

ADDC Add Accumulator with Carry / Añadir al Acumulador con Acarreo.

AJMP Absolute Jump / Salto Absoluto.

ANL Logical AND for byte variables / AND Lógico entre variables tipo byte.

ANL bit Logical AND for bit variables / AND Lógico entre variables tipo bit.

CJNE Compare and Jump if Not Equal / Compara y Salta si No es Igual.

CLR A Clear Accumulator / Limpia el Acumulador.

CLR bit Clear bit / Limpia un bit.

CPL A Complement Accumulator / Complementa (NOT) el Acumulador.

CPL bit Complement bit / Complementa (NOT) un bit.

DA Decimal Adjust of Accumulator / Ajuste Decimal del Acumulador.

DEC Decrement Register / Decrementa un Registro.

DIV Divide Accumulator by B / Divide el Acumulador entre B.

---

[27] Wikipedia, la enciclopedia libre, (2021). *INTEL 8051*. Recuperado de https://es.wikipedia.org/wiki/Intel_8051

[28] Taller Electrónica Blog. (2021). *Set de instrucciones 8051*. Recuperado de https://tallerelectronica.com/set-instrucciones-8051/

DJNZ Decrement Register and Jump if Not Zero / Decrementa Registro y Salta si No es Cero.

INC Increment Register / Incremente un Registro.

JB Jump if Bit Set / Salta si Bit es igual a Uno.

JBC Jump if Bit Set and Clear Bit / Salta si Bit es Igual a Uno y lo pone a Cero.

JC Jump if Carry Set / Salta si el Bit de Acarreo es Igual a Uno.

JMP @ Jump indirect to Address / Salto indirecto a Dirección contenida en Variable.

JNB Jump if Bit Not Set / Salto si Bit es igual a Cero (No igual a Uno).

JNC Jump if Carry Not Set / Salto si Bit de Acarreo No es Igual a Cero.

JNZ Jump if Accumulator Not Zero / Salto si Acumulador No es Igual a Cero.

JZ Jump if Accumulator Zero / Salto si Acumulador es Igual a Cero.

LCALL Long Call / Llamada a Dirección Lejana.

LJMP Long Jump / Salto a Dirección Lejana.

MOV Move byte variable / Copia variable tipo Byte.

MOV bit Move bit / Copia variable tipo Bit.

MOVC Move Code Memory / Copia Byte de Memoria de Programa.

MOVX Move External Memory / Copia Byte de Memoria Externa.

MUL Multiply Accumulator by B / Multiplica el Acumualdor por B.

NOP No Operation / Sin Operación.

ORL Logical OR for byte variables / OR Lógico entre variables tipo byte.

ORL bit Logical OR for bit variables / OR Lógico entre variables tipo bit.

POP Pop From Stack / Recupera Byte de la Pila de Memoria.

PUSH Push Onto Stack / Guarda Byte en la Pila de Memoria.

RET Return From Subroutine / Retorna de Subrutina.

RETI Return From Interrupt / Retorna de Subrutina de Interrupción.

RL Rotate Accumulator Left / Rota Acumulador a la Izquierda.

RLC Rotate Accumulator Left Through Carry / Rota Acumulador a la Izquierda a través del bit de Acarreo.

RR Rotate Accumulator Right / Rota Acumulador a la Derecha.

RRC Rotate Accumulator Right Through Carry / Rota Acumulador a de Derecha a través del bit de Acarreo.

SETB Set Bit / Pone a Uno una variable de tipo Bit.

SJMP Short Jump / Salto a Dirección Cercana.

SUBB Subtract From Accumulator With Borrow / Resta de Acumulador con Acarreo.

SWAP Swap Accumulator Nibbles / Intercambio de Nibbles de Acumulador.

XCH Exchange Bytes / Intercambia contenido con Acumulador.

XCHD Exchange Digits / Intercambia contenido con Nibble bajo del Acumulador.

XRL Exclusive OR / OR Exclusiva Lógica (XOR) entre variables tipo byte.

2.23 ARCHIVO HEX DE INTEL

El Hexadecimal Object File Format es un formato de archivo para la programación de microcontroladores, EPROMs y otros circuitos integrados. Es uno de los formatos más antiguos con esta finalidad.

Consiste en un archivo de texto cuyas líneas contienen valores hexadecimales que codifican los datos, y su offset o dirección de memoria.

Los distintos tipos de Intel Hex (8-bit, 16-bit y 32-bit) se diferencian en su endianness.

Cada línea consta de los siguientes elementos:

1. Código de inicio, un símbolo ':'
2. Longitud del registro, dos dígitos hexadecimales con la cantidad de bytes del campo de datos. Usualmente son 16 o 32 bytes.
3. Dirección, cuatro dígitos hexadecimales en big endian, con la dirección de inicio de los datos. Para direcciones mayores a 0xFFFF se emplean otros tipos de registro.
4. Tipo de registro, dos dígitos hexadecimales, de 00 a 05, definen el tipo del campo de datos
5. Datos, duplas de dígitos hexadecimales que contienen los datos
6. Checksum, dos dígitos hexadecimales con el complemento a dos de la suma de todos los campos anteriores, salvo el ':'.

Hay seis tipos de registros:

- 00, Datos, contiene una dirección de 16 bits y los datos correspondientes
- 01, Fin de archivo, no contiene datos y debe estar al final del archivo.
- 02, Dirección Extendida de Segmento, dirección base del segmento, para acceder a direcciones con más de 16 bits. Este valor se desplaza 4 bits a la izquierda (= multiplicar con 16) y se suma a la dirección proporcionada por los registros de datos. Su campo de longitud debe valer 02 y el de dirección 0000.
- 03, Dirección de Comienzo de Segmento, especifica los valores iniciales de los registros CS:IP, para procesadores 80x86. El campo de dirección es 0000,

longitud 04 y los datos contienen dos bytes para el segmento de código y otros dos para el instruction pointer

- 04, Dirección Lineal Extendida, permite dirigirse a 32 bits de memoria al contener los 16 bits superiores de la dirección. Su campo de dirección vale 0000 y el de longitud 02.
- 05, Comienzo de Dirección Lineal. Contiene 4 bytes que se cargan en el registro EIP de los procesadores 80386 y superiores. Su campo de dirección vale 0000 y el de longitud 04.

Existen varios sub-formatos:

- I8HEX o INTEL 8, de 8 bits
- I16HEX o INTEL 16, de 16 bits. Emplea registros 02 y 03, y la endianness de los datos puede variar.
- I32HEX o INTEL 32, de 32 bits. Agrega los registros 04 y 05.

Los procesadores Motorola utilizan un formato similar, denominado SREC.[29]

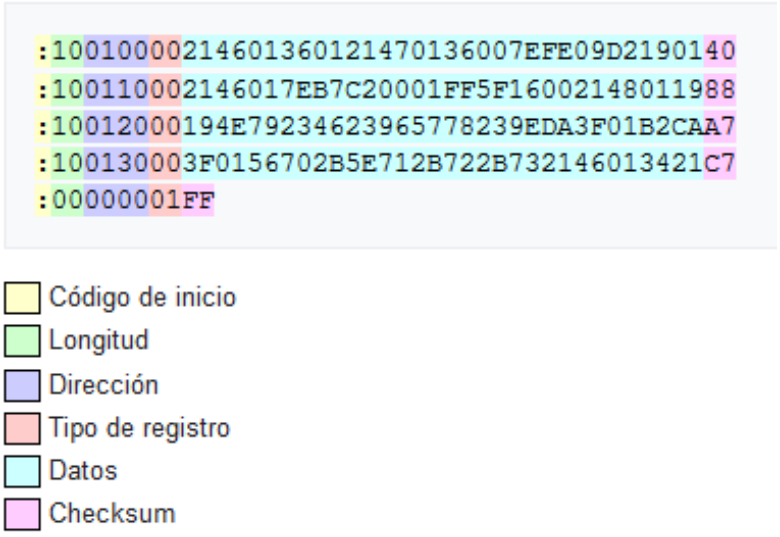


Figura 28: Ejemplo de formato de archivo HEX de INTEL.

Fuente: https://es.wikipedia.org/wiki/HEX_(Intel)

[29] Wikipedia, la enciclopedia libre, (2021). *HEX (INTEL).* Recuperado de https://es.wikipedia.org/wiki/HEX_(Intel)

2.24 TARJETAS DE DESARROLLO SIMILARES

en la actualidad existen diversos sistemas embebidos que diferentes microcontroladores o microprocesadores, los cuales están disponibles en el mercado, la mayoría de estos son de software y hardware libre y cuentan con librerías y código de ejemplos en internet.

Entre los más conocidos esta Arduino, una tarjeta de desarrollo que usa un microcontrolador de 8 bits de la serie ATMEGA de ATMEL.

Se encuentra también BASIC STAMP, un arreglo que contiene un microcontrolador PIC de 8 bits y usa por lenguaje de programación BASIC.

Y por último se hace mención a RaspBerry Pi, una placa de desarrollo que contiene un microprocesador de 32 bits de la familia ARM, el cual cuenta con un sistema operativo que puede ser Linux o Windows IoT.

2.24.1 COMPARACIÓN DE TARJETAS DE DESARROLLO

Las desventajas de los mencionados sistemas embebidos respecto al sistema propuesto son los siguientes:

TARJETA DE DESARROLLO	**VENTAJAS**	**DESVENTAJAS**
MCS-51 (sistema propuesto)	Ampliable en memoria de trabajo y de programa Ampliable en interfaces que pueden ser mapeadas en un bus de direcciones de 16 bits y un bus de datos de 8 bits Puede usarse cualquier compilador o ensamblador que soporte la familia de microcontroladores MCS-51. Destinado a proyectos que pueden crecer al pasar del tiempo tal y como se haría con un microprocesador. Cuenta con un puerto serie el cual sirve para cargar los programas, pero también para trabajar enviando o recibiendo datos que el usuario requiera.	Se requiere mayores aptitudes en programación y conocimiento de los registros. No existen librerías en la red, por lo que si se desea simplificar cierto proceso; el ususario deberá diseñar su propia librería. Puede tener interferecias si se usa un bus demasiado largo para interconectar con otras interfaces. El consumo de energía aumenta considerablemente al aumentar las diferentes placas de interfaces ya que se debe alimentar a muchos integrados.
Arduino	Bastante información y códigos de ejemplo en internet. Diferentes modelos que varía de acuerdo a los precios de	No es posible expandir la memria de trabajo o de programa. No es posible acceder a todos los

	adquisición. Programación intuitiva y llevadera, (no hay necesidad de ser un experto programador). Diferentes módulos de expansión o de facilitación de realización de proyectos. (no hay necesidad de tener muchos conocimientos en electrónica) Conector USB para enviar alimentar y cargar los programas (no necesita fuente externa)	registros específicos del microcontrolador principal, ya que muchos de estos trabajan por debajo para facilitar la programación como por ejemplo el temporizador y todos los registros dependientes de este. No es posible mapear interfaces de expansión ya que no cuenta con los buses de direccione, datos y control de manera externa. Al conectar ciertos módulos como ser para controlar motores y no se usa fuente externa el puerto USB del PC podría quedar inservible. Solo trabaja en su propio entorno de desarrollo integrado
Basic Stamp	Lenguaje de programación bastante llevadero. Ejemplos disponibles en la red.	Solo trabaja con su propio entorno de desarrollo integrado
RaspBerry Pi	Cuenta con puertos USB Host. Cuenta con puerto HDMI para salida de video. Cuenta con un sistema operativo.	Si bien es un microprocesador de 32 bits no es posible realizar el mapeo de interfaces o memorias ya que no cuenta con los buses respectivos. Solo tiene un puerto digital para trabajar. Todo se almacena en una memoria SD, y es susceptible a dañarse y borrar tanto los programas como sus propio sistema

Tabla 9: Tabla de comparación de las principales tarjetas o placas de desarrollo.

Fuente: Diseño propio.

CAPITULO III
DESARROLLO DEL PROYECTO

En ésta etapa se planteará el diseño de la placa de desarrollo, usando los conceptos ya mencionados para su construcción.

3.1 DESCRIPCIÓN GENERAL DEL SISTEMA

El sistema cuenta principalmente con un microcontrolador AT89C52 el cual se encargará de gestionar los programas y enviarlos a una memoria externa para después ejecutarlos. La comunicación, para la transferencia de datos entre la PC y la tarjeta de desarrollo se efectúa mediante puerto serie, el mismo que está integrado al microcontrolador principal. También se cuenta con un circuito que intercambia la memoria interna por la externa en el microcontrolador usando un multiplexor/demultiplexor. Por último, se tiene un bus de expansión para poder conectar periféricos o memorias, y también se debe resaltar que la alimentación debe hacerse con una fuente externa de 5 V ya que la tarjeta no cuenta con una fuente incorporada.

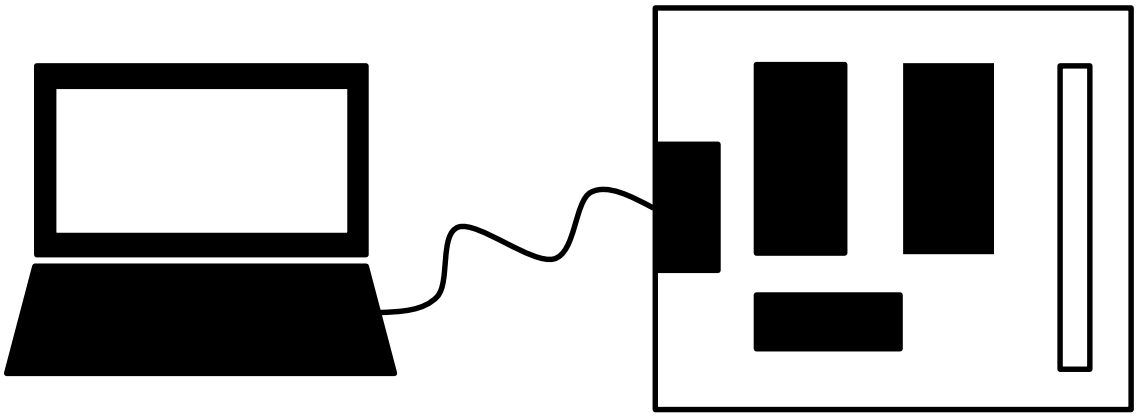


Figura 29: esquema básico de la conexión de la tarjeta de desarrollo.

Fuente: Diseño propio

3.2 DISEÑO DEL BOOTLOADER PARA EL MICROCONTROLADOR

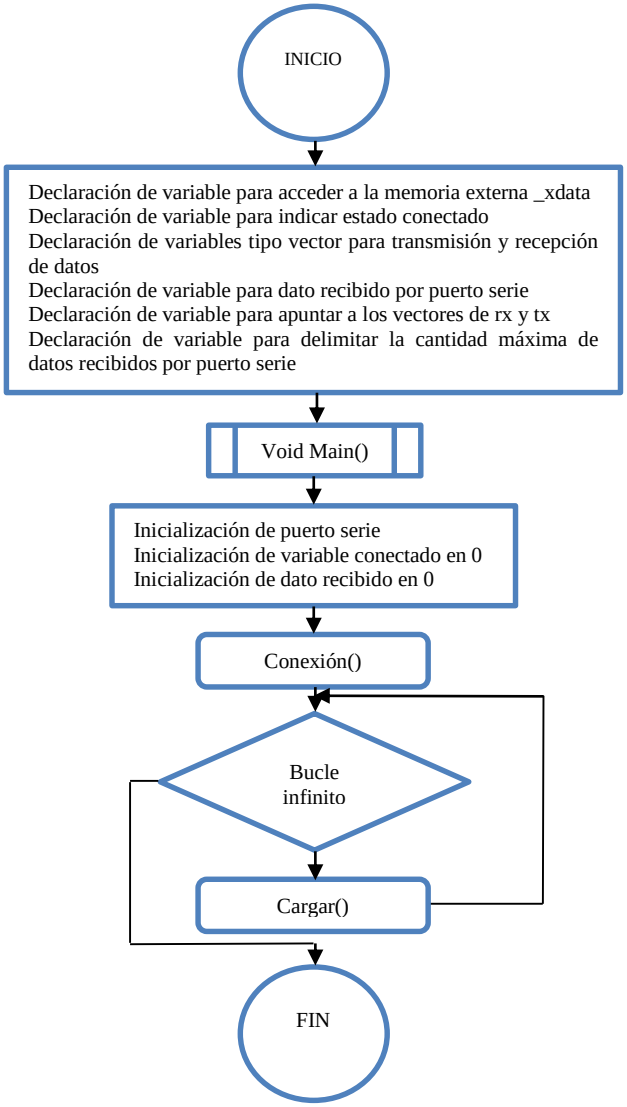


Figura 30: Diagrama de flujo del bootloader del microcontrolador parte 1.

Fuente: Diseño propio.

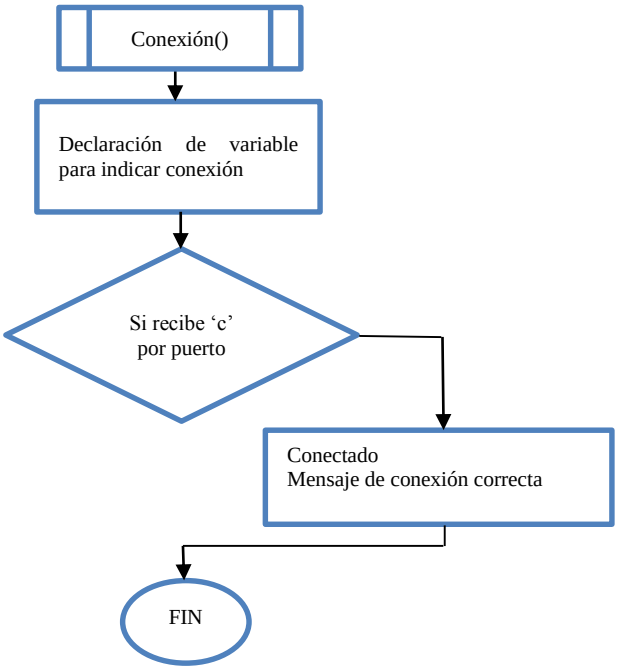


Figura 31: Diagrama de flujo del bootloader del microcontrolador parte 2.

Fuente: Diseño propio.

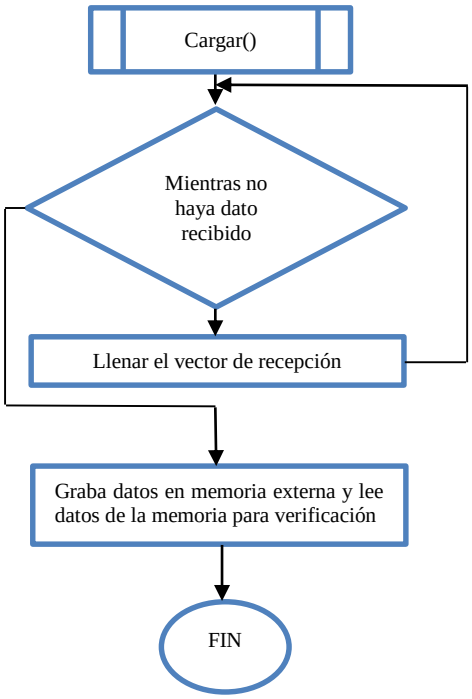


Figura 32: Diagrama de flujo del bootloader del microcontrolador parte 3.

Fuente: Diseño propio.

El diagrama de flujo anterior muestra cómo se cargan los datos en la memoria externa, esto a partir de un archivo HEX de Intel, interpretado por el cargador de archivos compilados y enviado hacia el microcontrolador con el siguiente formato:

NN	DDDD	XX	XX	XX	XX	XX	XX	XX	XX

Donde:

- NN: Es la cantidad de datos a grabar, el valor mínimo es 01H y el máximo 20H.
- DDDD: Es la dirección de la memoria externa que puede variar desde 0000H hasta FFFFH.
- XX: Son los datos a grabar, estos son de un byte lo que quiere decir que pueden variar desde 00H hasta FFH.

3.3 DESARROLLO DE LA APLICACIÓN PARA CARGAR ARCHIVOS COMPILADOS HEX

La aplicación tendrá un entorno GUI (Interfaz gráfica de usuario) la cual se desarrolla bajo el lenguaje de Visual Basic.NET, que pertenece a Visual Studio.NET, para lo cual se iniciará un nuevo proyecto.

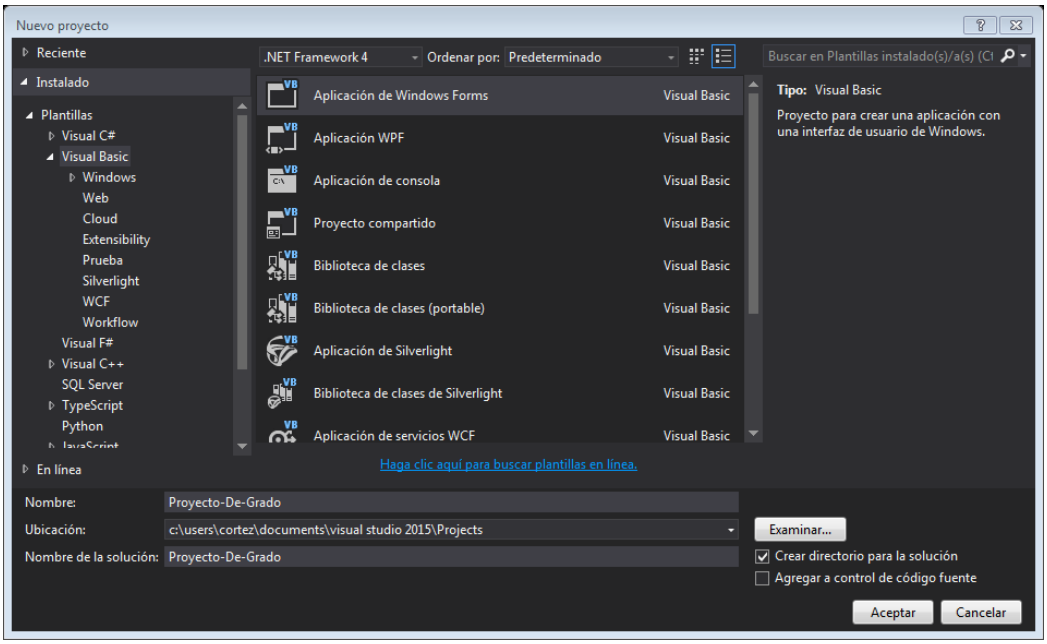


Figura 33: Creación de un nuevo proyecto en Visual Studio.NET.

Fuente: Captura de pantalla.

Una vez creado el proyecto se mostrará el formulario principal en donde se pondrán los controles.

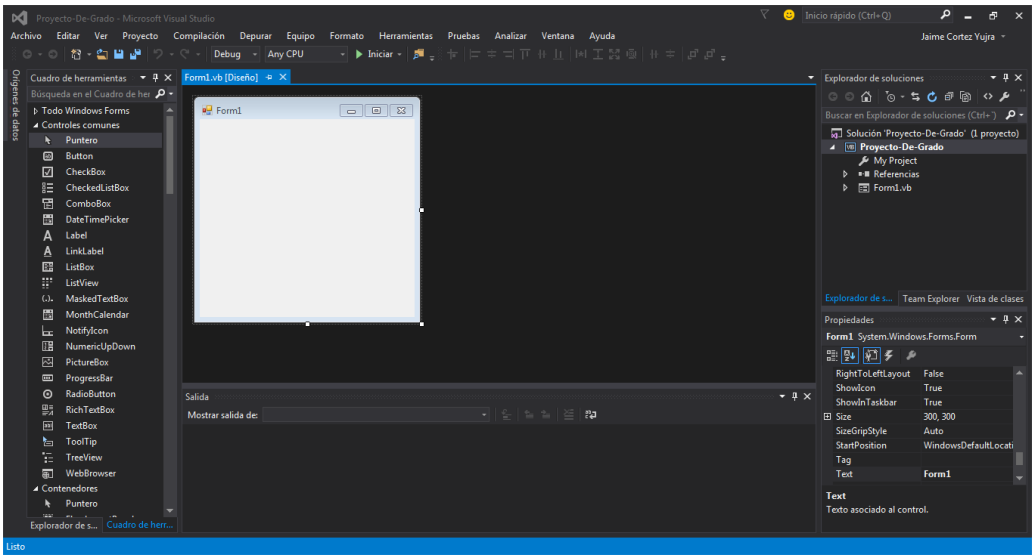


Figura 34: Vista del formulario.

Fuente: Captura de pantalla.

Después de agregar los controles el formulario se ve de la siguiente manera.

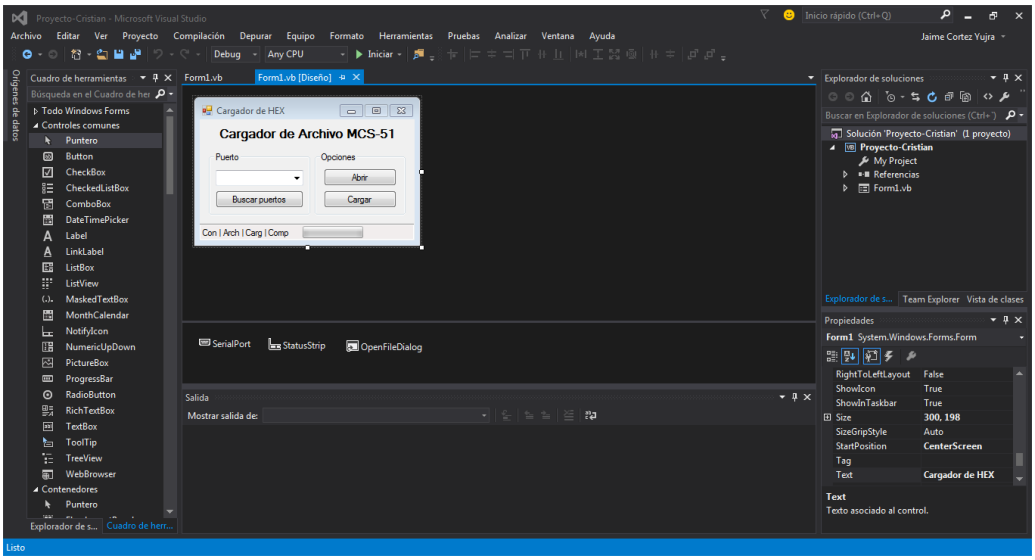


Figura 35: Vista del formulario con controles.

Fuente: Captura de pantalla.

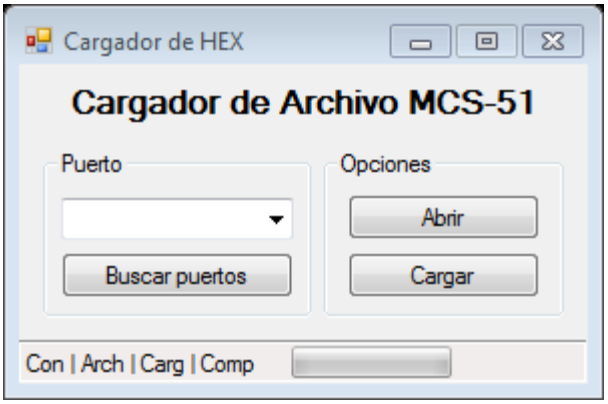


Figura 36: Vista del formulario en ejecución.

Fuente: Captura de pantalla.

El programa consiste en seleccionar un puerto disponible en el cual esté conectado la tarjeta de desarrollo, abrir un archivo HEX y por ultimo cargarlo al microcontrolador y este a su vez a la memoria externa del mismo.

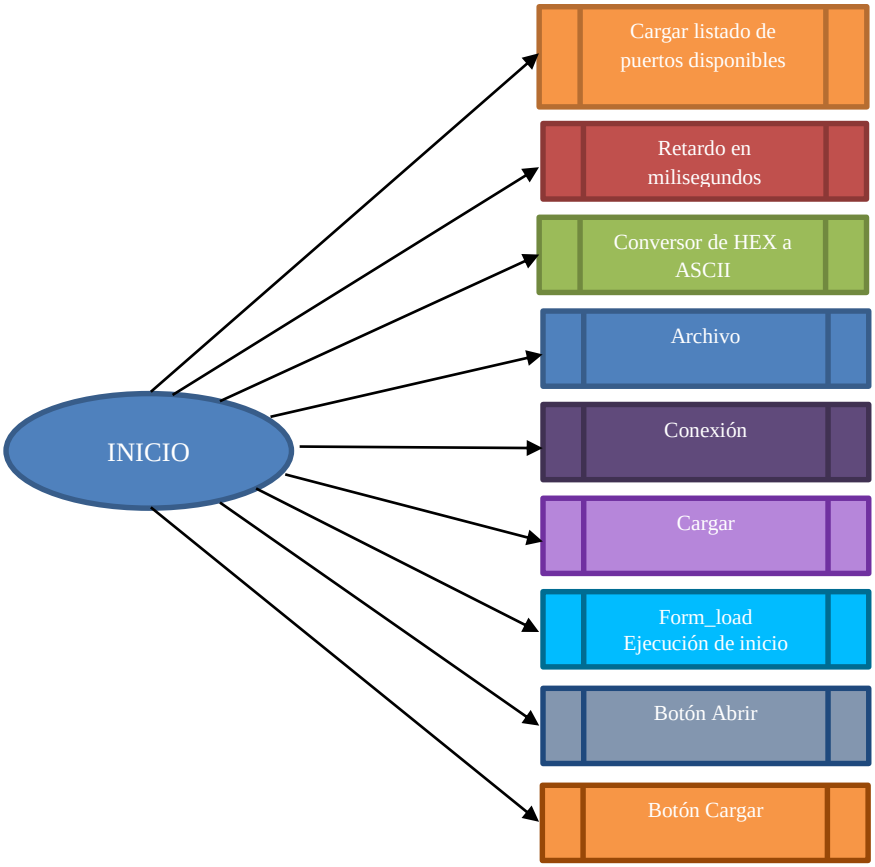


Figura 37: Diagrama de flujo general del programa de carga de archivos HEX.

Fuente: Diseño propio.

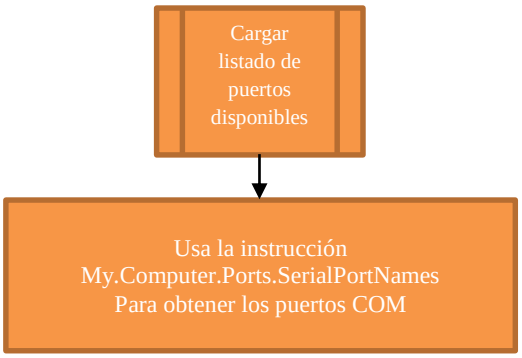


Figura 38: Diagrama de flujo del subprograma para cargar listado de puertos.

Fuente: Diseño propio.

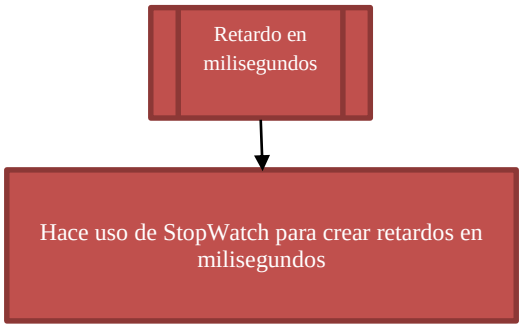


Figura 39: Diagrama de flujo de la función para retardos.

Fuente: Diseño propio.

Figura 40: Diagrama de flujo de la función para convertir HEX a ASCII.

Fuente: Diseño propio.

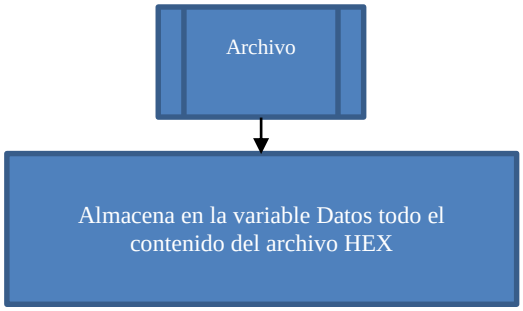


Figura 41: Diagrama de flujo del subprograma para leer archivo.

Fuente: Diseño propio.

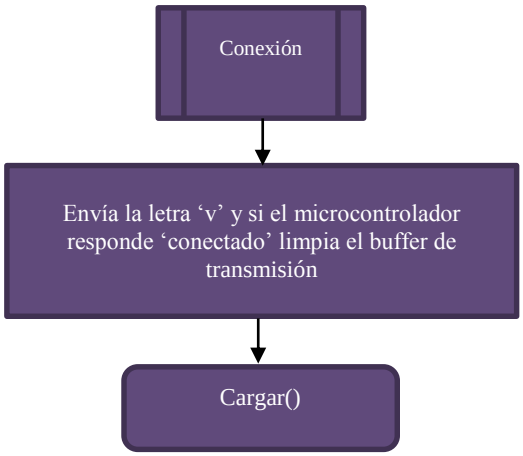


Figura 42: Diagrama de flujo del subprograma para conexión.

Fuente: Diseño propio.

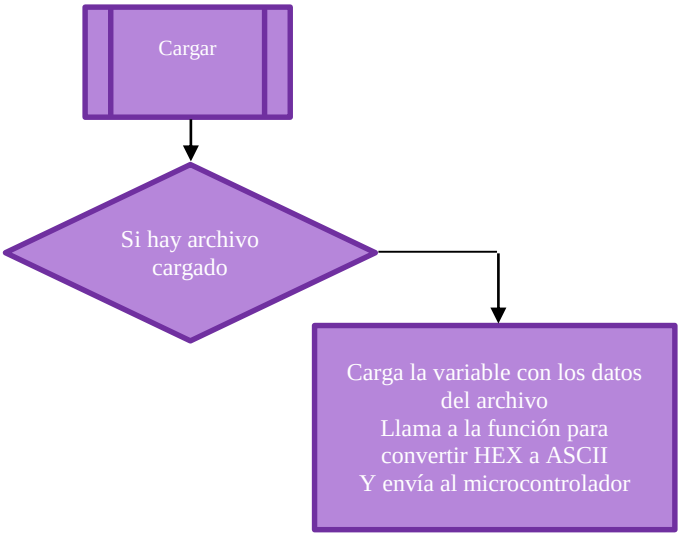


Figura 43: Diagrama de flujo del subprograma para cargar.

Fuente: Diseño propio.

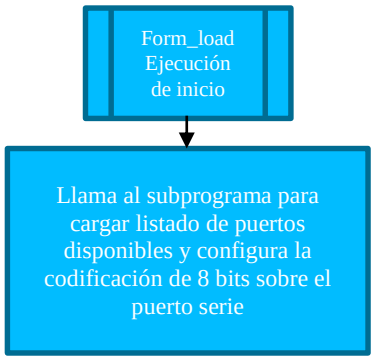


Figura 44: Diagrama de flujo para la carga del formulario.

Fuente: Diseño propio.

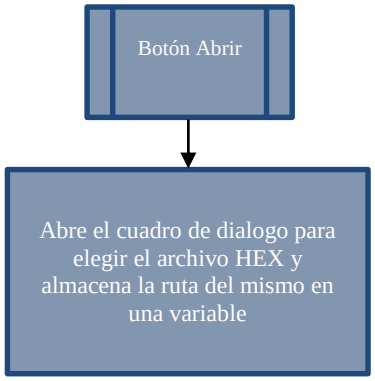


Figura 45: Diagrama de flujo para el botón abrir.

Fuente: Diseño propio.

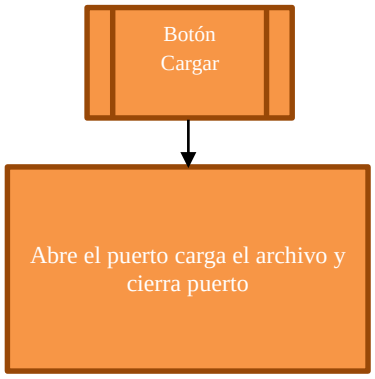


Figura 46: Diagrama de flujo para el botón cargar.

Fuente: Diseño propio.

3.4 DISEÑO DE LA PLACA DE DESARROLLO

El diseño consta en intercambiar la memoria externa de programa con la memoria externa de datos (ROM por RAM), usando un MUX/DEMUX 4053 el cual se encargará de ese trabajo y de un MAX232 el cual activará ese MUX mediante el pin de RST del puerto. El segundo MAX232, se encargará de las transmisión y recepción de datos.

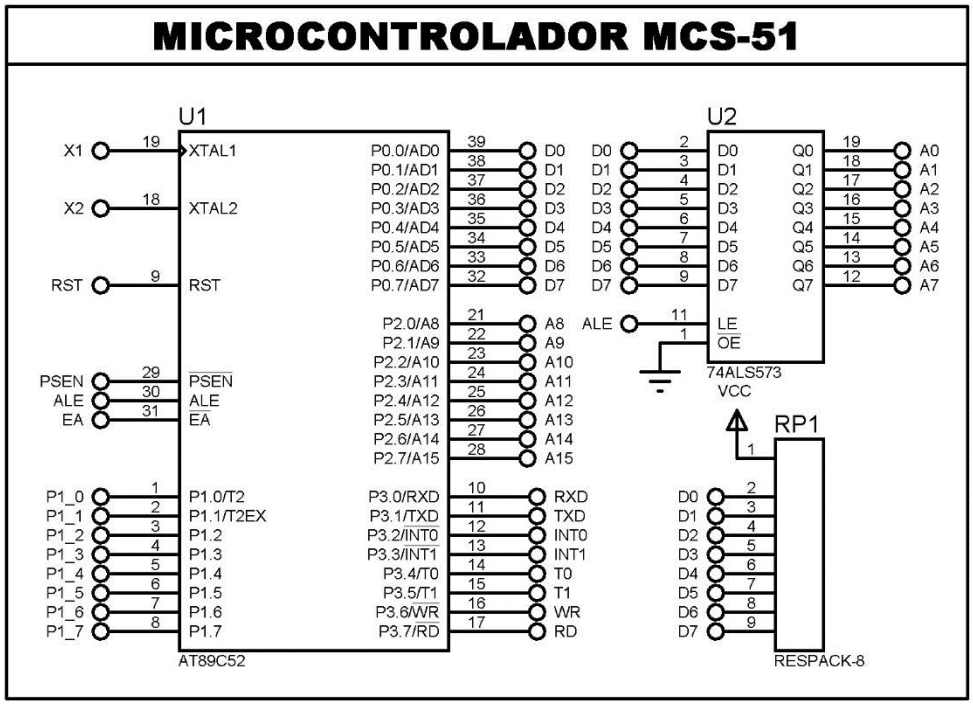


Figura 47: Esquema microcontrolador, Latch D y resistencias para el bus de datos.

Fuente: Captura de pantalla.

Como se observa el microcontrolador principal tiene un Latch D para separar direcciones bajas de los datos ya que es un bus compartido, el pin ALE es el encargado de realizar la separación activando o desactivando el Latch D. también se cuenta con un empaque de resistencias ya que el bus de datos tiene solo los estados bajo y latente, por lo que se hace necesario este empaque y dará un estado alto cuando el microcontrolador entregue un estado latente.

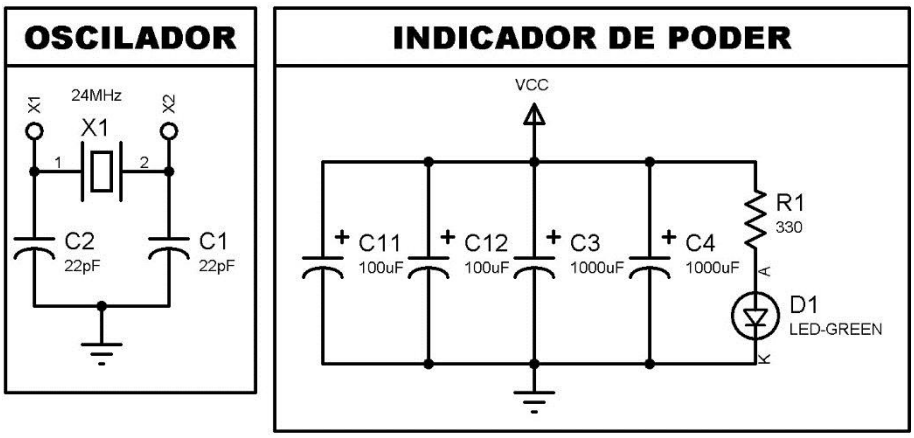


Figura 48: Esquema oscilador e indicador de poder.

Fuente: Captura de pantalla.

Como se observa, se hará uso de un cristal de 24 MHz, para que el microcontrolador tenga una frecuencia de trabajo de 2 MHz, también se implementa un led para indicar la alimentación de la tarjeta y capacitores que filtraran ruidos en la alimentación.

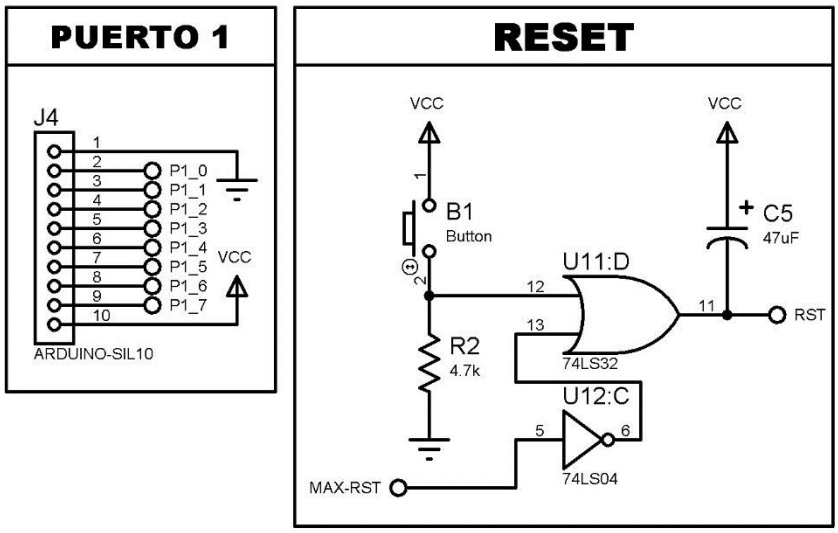


Figura 49: Esquema Puerto 1 y etapa de reset.

Fuente: Captura de pantalla.

Se implementa una salida para el uso del puerto disponible en el microcontrolador y también la etapa de reset que será de manera manual por un pulsador y también mediante el puerto serie controlado por la PC.

Button	MAX-RST	RST
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**1**

Tabla 10: Tabla de verdad de la compuerta OR correspondiente a la etapa de RESET.

Fuente: Diseño propio.

Como se observa en la figura el circuito hace un reset al microcontrolador cuando button es igual a 1 o cuando max-rst es igual a 0.

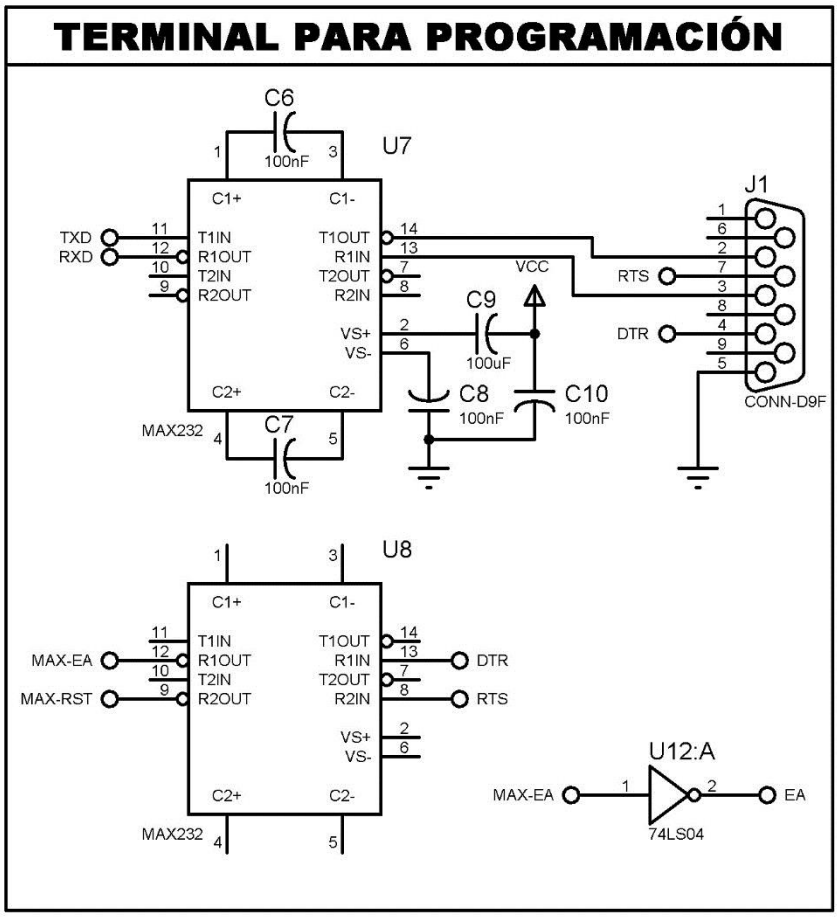


Figura 50: Esquema terminal de comunicación.

Fuente: Captura de pantalla.

Para la terminal se usan dos drivers uno encargado de la comunicación como tal y el otro encargado de alternar la memoria de datos con la memoria de programa para poder hacer la transferencia de un programa, además de poder reiniciar el microcontrolador.

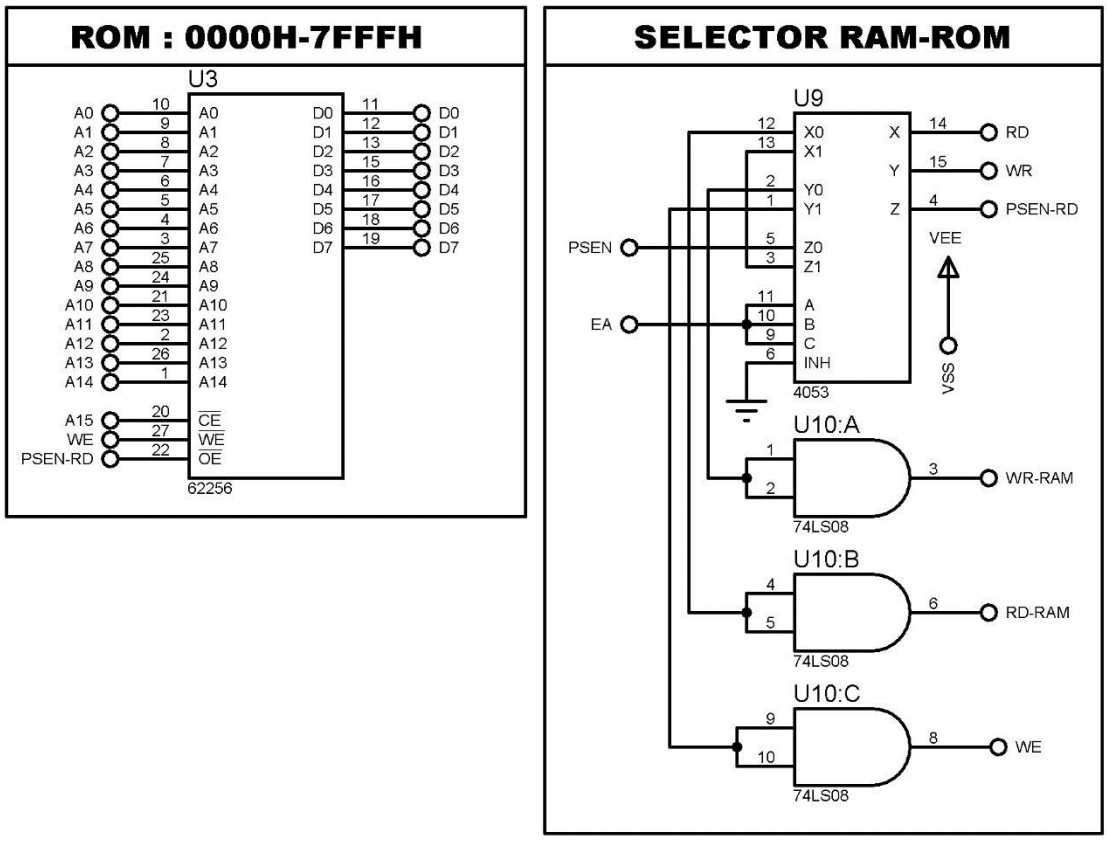


Figura 51: Esquema memoria de programa y selector de memoria.

Fuente: Captura de pantalla.

En esta etapa se muestra que se tiene una memoria RAM pero que sobre la misma se cargara el nuevo programa funcionando esta como una memoria "ROM" ya que la transferencia de datos es mucho más rápida, también se observa que mediante compuertas AND y el MUX se ha implementado el intercambio de espacios de datos por espacios de programa en las líneas RD WR y PSEN.

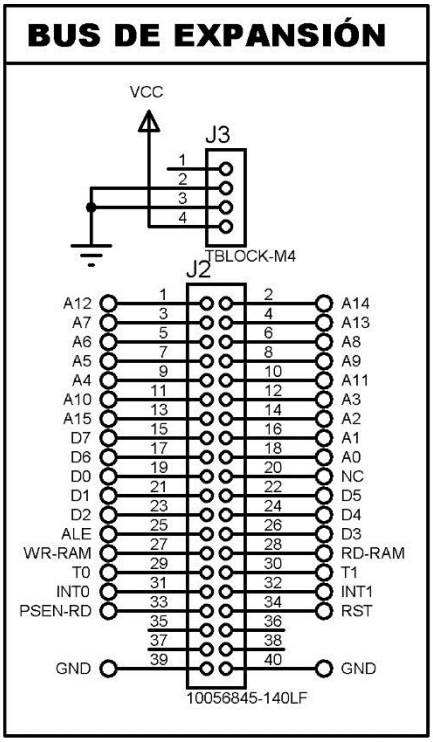


Figura 52: Esquema bus de expansión y terminal de alimentación.

Fuente: Captura de pantalla.

Finalmente se tiene un bus donde se podrán conectar más interfaces o memorias para ampliar la capacidad del microcontrolador, véase que este bus tiene tanto las líneas de datos, direcciones y de control, además de otras líneas extra como interrupciones externas y temporizador.

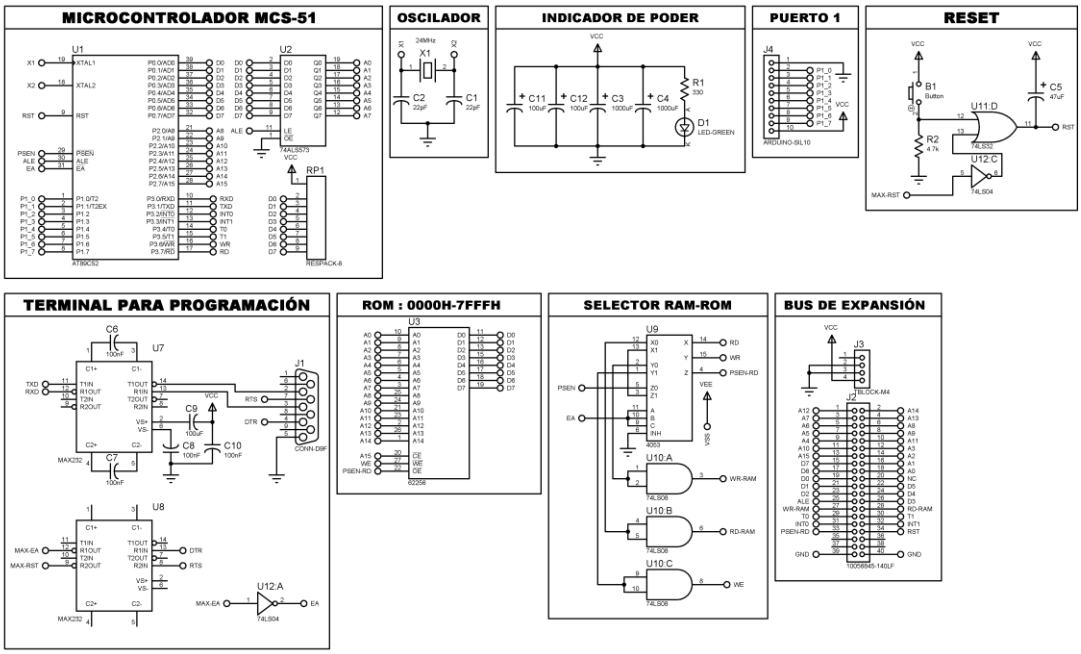


Figura 53: Esquema completo de la placa de desarrollo.

Fuente: Captura de pantalla.

3.5 CARACTERÍSTICAS PRINCIPALES DE LA PLACA DE DESARROLLO

Las caracteristicas principales son similares a las de un AT89C52 ya que es el microcontrolador que se usará, a continuacion se detallan algunas.

Caracteristica	**Minimo**	**Maximo**
Temperatura de operacion	-55 º	125 º
Voltaje en pin con respecto de tierra	-1 V	7 V
Oscilador		24 MHz
Puerto serie integrado		1
Voltaje de grabado	11,5 V	12,5 V
Voltaje de operación	4,5 V	5,5 V

Tabla 11: Tabla de las principales características de la placa de desarrollo.

Fuente: Diseño propio.

3.6 EJEMPLOS DE APLICACIÓN

3.6.1 GENERACIÓN DE ONDA CUADRADA

En un sistema es frecuente tener que generar una señal periódica de frecuencia determinada y lo más simétrica posible. En este ejemplo, se propone la creación de una señal cuadrada de 5 kHz de frecuencia, que se extraerá por la patilla P1.0 de un microcontrolador 8051 o 8052.

Para generar una frecuencia de 5kHz se precisa de un periodo de 200/s y, por tanto, cambiar el estado de la patilla P1.0 cada 100/s. Para cambiar el estado de P1.0 se utilizará la instrucción CPL de complemento de un bit, y para generar un retardo de 100/s se utilizará una rutina de retardo.

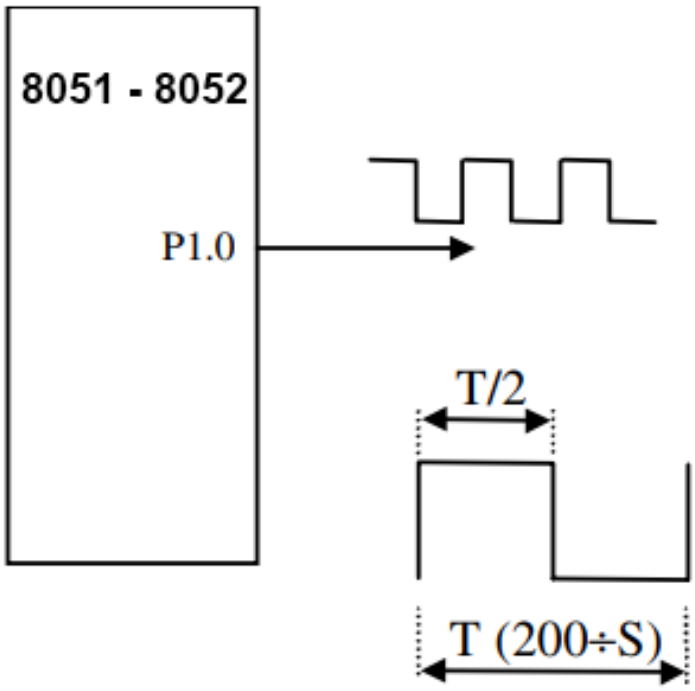


Figura 54: Esquema básico de para la generación de onda cuadrada.

Fuente: Microcontroladores MCS-51 y MCS-251 – José Matas Alcalá, Rafael Ramón Ramos Lara (Modificado).

```
;*********************************************************
; Programa de generación de una señal cuadrada
;*********************************************************
ORG 0H
LJMP Principal
;*********************************************************
; Rutina Principal
;*********************************************************
ORG 0100H
Principal:   CPL P1.0 ;Complementa el estado lógico de P1.0
             CALL Retardo ;Llama a la rutina de retardo
             SJMP Principal ;Bucle infinito a principal
Retardo:     MOV R7, #46 ;Pone 46 en R7
             L1: DJNZ R7, L1 ;Bucle sobre L1
             RET ;Retorno de subrutina
;*********************************************************
```

3.6.2 CONEXIÓN DE UN DISPLAY DE 7 SEGMENTOS

Los dígitos de visualización se usan de manera frecuente para indicar al usuario una determinada información. Los dígitos suelen estar formados por 7 segmentos o por 16 segmentos (hexadecimales), aunque existen en el mercado una gran variedad y gama de dígitos a disposición del diseñador. Los dígitos de 7 segmentos son adecuados para visualizar números y algunas letras y símbolos, estos últimos con poca definición.

En este ejemplo se conectan tres teclas y un dígito de 7 segmentos al microcontrolador.

El programa deberá leer el estado de las teclas y poner en el dígito la letra “A” si se pulsa la tecla T1, la letra “b” si se pulsa la tecla T2, y la letra “C” si se pulsa la tecla T3.

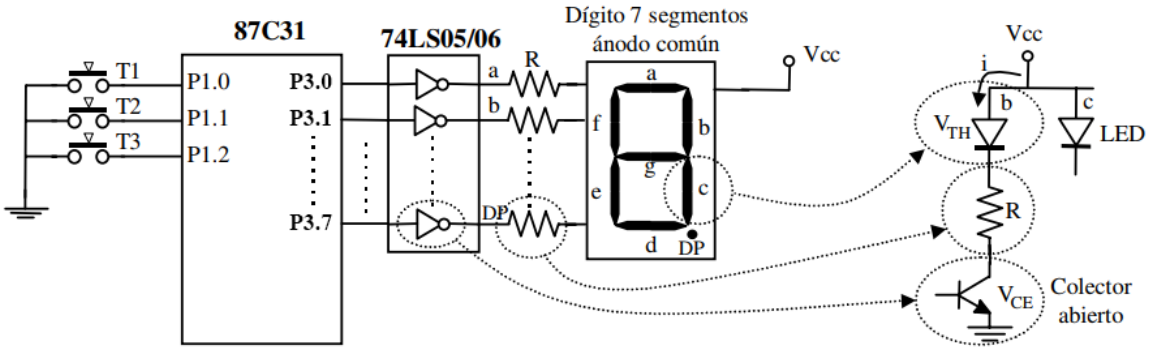


Figura 55: Esquema básico de conexión de display de 7 segmentos.

Fuente: Microcontroladores MCS-51 y MCS-251 – José Matas Alcalá, Rafael Ramón Ramos Lara (Modificado).

```
;*************************************************************
; Programa para la conexión de un dígito de 7-segmentos
;*************************************************************
ORG 0H
MOV P2, #0 ;Apaga todos los leds del dígito
LJMP Principal
;*************************************************************
; Rutina Principal
;*************************************************************
ORG 0100H
Principal:   JNB P1.0, Tecla_T1 ;Comprueba si se ha pulsado T1
             JNB P1.1, Tecla_T2 ;Comprueba si se ha pulsado T2
             JNB P2.0, Tecla_T3 ;Comprueba si se ha pulsado T3
             MOV P3, #0 ;Borra el dígito
             SJMP Principal
Tecla_T1:    MOV P3, #0111 0111b ;Pone letra A en el dígito
             SJMP Principal ;Regresa a Principal
Tecla_T2:    MOV P3, #0111 1100b ;Pone la letra b en el dígito
             SJMP Principal ;Regresa a Principal
Tecla_T3:    MOV P3, #0011 1001b ;Pone la letra C en el dígito
             SJMP Principal ;Regresa a Principal
;*************************************************************
```

3.6.3 ACCESO A DIRECCIONES PARA LECTURA Y ESCRITURA

Se muestra el código base en SDCC para escribir y leer datos de 8 bits en direcciones de 16 bits.

```
__xdata unsigned char * __data p;      //Variable de tipo puntero externo
void  Escribir(unsigned  int  Dir,  unsigned  char  Dato)  //Procedimiento  de
escritura
{
   p[Dir] = Dato;
}
unsigned char Leer(unsigned int Dir) //Funcion de lectura
{
   return p[Dir];
}
```

CAPITULO IV

ANÁLISIS DE COSTOS

4.1 COSTO DE MATERIALES

TIPO	CANTIDAD	DETALLE	VALOR	PRECIO UNITARIO EN BS.
Resistores	1	¼ de vatio	330 Ω	0.20
	1	¼ de vatio	4,4 KΩ	0.20
Capacitores	2	Electrolítico	1000 uF	1.5
	2	Electrolítico	100 uF	1
	1	Electrolítico	47 uF	1
	5	Cerámico	100 nF	1
	2	Cerámico	22 pF	1
Circuitos Integrados	2	Driver	MAX232	10
	5	TTL 74 AND OR	-	5
	1	PPI	8255	20
	1	MUX-DEMUX	CD4053	4
	1	Microcontrolador	AT89C52	20
	1	LATCH D	74LS573	6
Conectores	1	RS-232 Hembra	DE-9 H	6
	1	40 pines Macho	SIL40 M	4
	1	USB - RS232 Cable	-	40
	1	40 pines Macho Doblado	SIL40 M	4
	2	2 terminales	Bornera 2	2.50
Otros	1	Placa PCB	-	20
TOTAL				**187.4**

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Los objetivos trazados inicialmente fueron alcanzados, de manera que se puede concluir que:

Se logró diseñar e implementar una tarjeta de desarrollo modular basada en la familia de microcontroladores mcs-51 para el desarrollo de múltiples aplicaciones.

- Se logró determinar las características técnicas y económicas del sistema desarrollado, mostrando los resultados en los apartados de diseño y costos.
- Se pudo diseñar la tarjeta de desarrollo propuesta, haciendo uso de Proteus como entorno para la creación de la parte esquemática como en el diseño de la placa.
- Se diseñó el software para cargar los programas a la tarjeta, realizado en Visual Basic. NET.
- Se implementó el prototipo final corrigiendo algunos errores del sistema.

5.2 RECOMENDACIONES

La tarjeta elaborada puede tener múltiples aplicaciones tanto didácticas como profesionales, pero deben tomarse en cuenta las siguientes recomendaciones:

Tener en cuenta que la corriente entregada por cada pin del microcontrolador o periférico es baja por lo que en su mayoría se requerirá hacer el uso de “pullup” con resistores o usar “dirvers” para aumentar la corriente de salida.

Las aplicaciones se cargarán en la memoria externa, por lo que se debe tener eso en cuenta a la hora de diseñar el programa.

Si se desea implementar un nuevo módulo, se debe tener muy en cuenta el mapeo del componente sobre una dirección correcta, y usar esa misma dirección a la hora de hacer el programa de control.

BIBLIOGRAFÍA

BIRTLH. (2021). *ELEC02.- Circuitos combinacionales MSI*. Obtenido de https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_54_decodificadores.html

BIRTLH. (2021). *ELEC02.- Circuitos combinacionales MSI*. Obtenido de https://ikastaroak.ulhi.net/edu/es/IEA/ELEC/ELEC02/es_IEA_ELEC02_Contenidos/website_53_codificadores.html

Desarrollo Web. (2021). *Tipos de datos en .NET*. Obtenido de https://desarrolloweb.com/articulos/1388.php

Fidias G. Arias. (1999). Concepto de Investigación. En *El Proyecto de Investigación. Guía para su elaboración* (pág. 22). Caracas: Episteme. Orial Ediciones.

Fidias G. Arias. (1999). Marco Metodológico. En *El Proyecto de Investigación. Guía para su elaboración* (pág. 45). Caracas: Episteme. Orial Ediciones.

Hardware Libre. (2021). *¿Qué es una placa SBC?* Obtenido de https://www.hwlibre.com/que-es-una-placa-sbc/

HETPRO, Herramientas Tecnológicas Profesionales. (2021). *Compuertas Lógicas*. Obtenido de https://hetpro-store.com/TUTORIALES/compuertas-logicas/

rjconcepcion. (2021). *Tarjetas de desarrollo (Episodio #8)*. Obtenido de https://www.rjconcepcion.com/podcast/tarjetas-de-desarrollo-episodio-8/

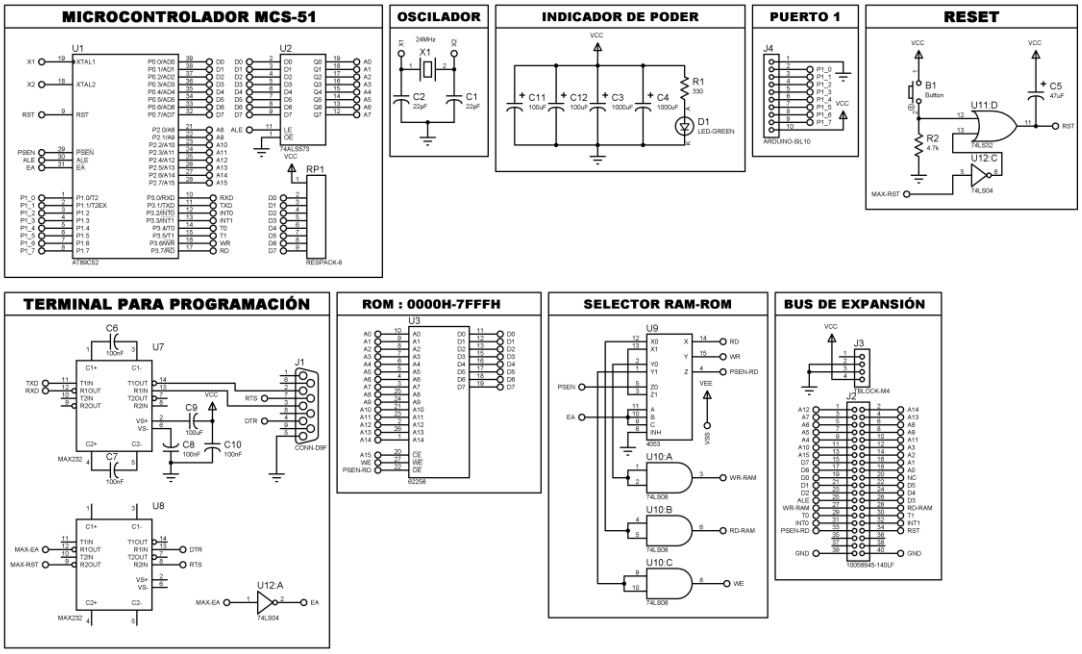
Taller Electrónica Blog. (2021). *Set de instrucciones 8051*. Obtenido de https://tallerelectronica.com/set-instrucciones-8051/

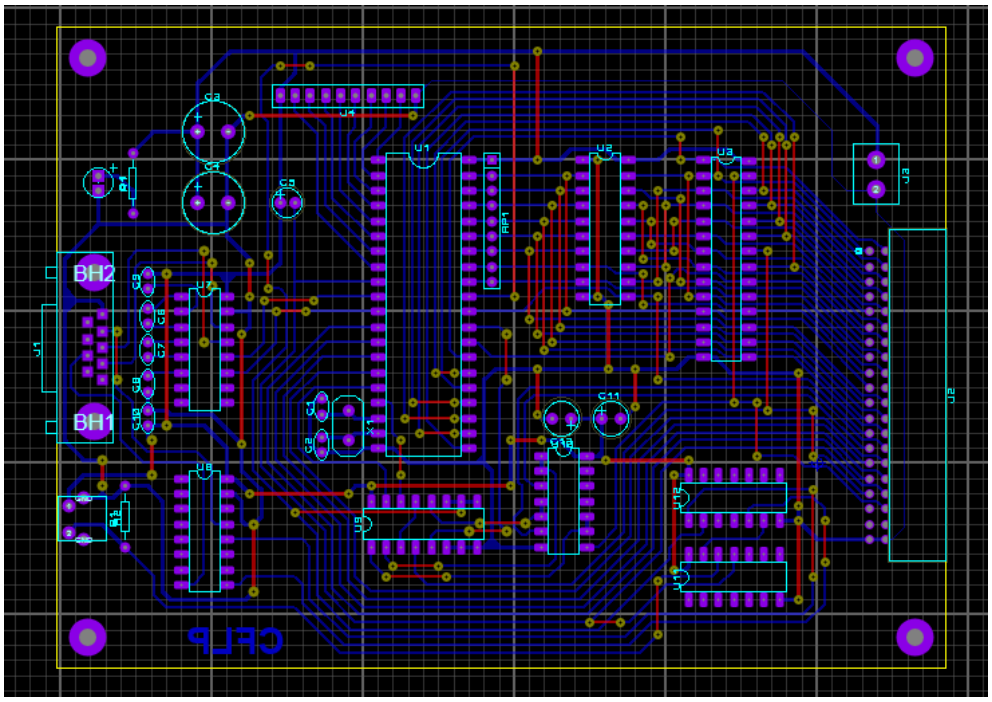
Wikipedia, la enciclopedia libre. (2021). *Memoria de acceso aleatorio*. Obtenido de https://es.wikipedia.org/wiki/Memoria_de_acceso_aleatorio

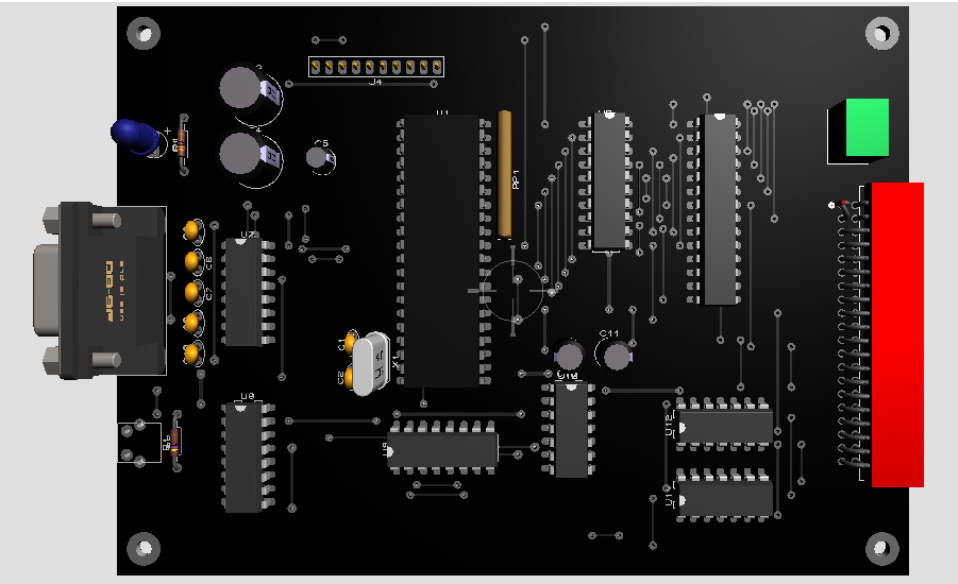
Wikipedia, la enciclopedia libre. (2021). *Arquitectura de Von Neumann*. Obtenido de https://es.wikipedia.org/wiki/Arquitectura_de_Von_Neumann

ANEXOS

ESQUEMA, PCB Y VISUALIZACIÓN 3D DE LA TARJETA DE DESARROLLO







CÓDIGO FUENTE DE LA APLICACIÓN PARA CARGAR ARCHIVOS COMPILADOS HEX

```
Public Class Form1
    Public Ruta As String               'Contendrá la ruta del archivo Hex
    Public Datos As String              'Contendrá toda la información del archivo abierto
    Public Array_Datos() As String      'Contendrá los datos linea a linea del archivo abierto
    Public Datos_Validos As String      'Contendrá los datos validos para la carga a memoria
    Public Lineas As UInteger           'Contendrá la cantidad de lineas del archivo abierto
    Public Verificar As String          'Contendrá los datos validos de la última linea cargada
    Public BuferRX As String            'Contendra los datos recibidos de la última linea cargada
    Public Archivo_Valido As Boolean    'Es verdadero si se ha cargado un archivo
    Public Cargando As Boolean          'Es verdadero si se está cargando los datos en memoria
    Public Estado(4) As String          'Contendrá los estados mostrados en la barra inferior

    Sub GetSerialPortNames()
        Dim Contador As Integer = 0
        ComboBox1.Items.Clear()                                 'Borra todos los items que
pudieran estar cargados
        For Each sp As String In My.Computer.Ports.SerialPortNames  'Obtiene los puertos COM
disponibles
            ComboBox1.Items.Add(sp)                             'Agrega los puertos encontrados al
ComboBox
        Next
        Contador = ComboBox1.Items.Count                        'Contador obtiene la cantidad de
items cargados
        If Contador = 0 Then                                        'Si la cantidad es cero
            ComboBox1.Items.Clear()                             'Borra todos los items que
pudieran estar cargados
            ComboBox1.Items.Add("vacio")                        'Agrega un item indicando que no
hay puertos
        Else                                                        'Caso contrario
            ComboBox1.SelectedIndex = 0                         'Selecciona el primer item de los
puertos encontrados
        End If
    End Sub

    Public Sub Delay_ms(ByVal interval As Integer)
        Dim sw As New Stopwatch
        sw.Start()
        Do While sw.ElapsedMilliseconds < interval
```

```
            Application.DoEvents()
        Loop
        sw.Stop()
    End Sub

    Public Function HexToAsc(ByVal HexText As String) As String
        Dim Valor As String
        Dim Texto As String
        Dim N As UInteger
        Verificar = ""
        Texto = ""
        Valor = ""
        N = Len(HexText)
        For i As Integer = 1 To N Step 2
            Texto = Mid(HexText, i, 2)
            Valor = Valor & Chr(Val("&H" & Texto))
        Next
        Verificar = Mid(Valor, 4)
        HexToAsc = Valor
    End Function

    Public Sub Archivo()
        Datos = ""
        Dim Delimitadores() As String = {vbCrLf, vbCr, vbLf}
        If My.Computer.FileSystem.FileExists(Ruta) Then
            Datos = My.Computer.FileSystem.ReadAllText(Ruta)
            Array_Datos = Datos.Split(Delimitadores, StringSplitOptions.None)
            Lineas = Array_Datos.Length
        End If
    End Sub

    Sub Conexion()
        SerialPort.Write("v")
        Delay_ms(100)
        BuferRX = SerialPort.ReadExisting
        If BuferRX = "CONECTADO" & vbNewLine Then
            Estado(0) = "Con"
            ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & "
| " & Estado(3)
            SerialPort.DiscardInBuffer()
            BuferRX = ""
            Cargar()
        Else
            MsgBox("No se ha podido conectar." & vbNewLine & "Verifique que el cable esté
conectado.", MsgBoxStyle.Critical + vbOKOnly, "Error de Conexión")
        End If
    End Sub

    Public Sub Cargar()
        Estado(2) = "Carg"
        ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & " | "
& Estado(3)
        For i As UInteger = 0 To Lineas - 2
            If Mid(Array_Datos(i), 1, 1) = ":" And Array_Datos(i) <> ":00000001FF" And
Mid(Array_Datos(i), 1, 1) <> ";" Then
                If Mid(Array_Datos(i), 8, 2) = "00" Then
                    Datos_Validos = Mid(Array_Datos(i), 2, 6) & Mid(Array_Datos(i), 10, 2 *
Val("&H" & Mid(Array_Datos(i), 2, 2)))
                    SerialPort.Write(HexToAsc(Datos_Validos))
                    Delay_ms(350)
                    BuferRX = SerialPort.ReadExisting
                    If Verificar <> BuferRX Then
                        MsgBox("Se ha verificado un código diferente al cargado." & vbNewLine &
"Linea: " & i + 1 & "del archivo cargado.", MsgBoxStyle.Critical + vbOKOnly, "Error de
Verificación")
                        Exit Sub
                    End If
                    ToolStripProgressBar.Value = ((i + 1) * 100 / (Lineas - 2))
                End If
```

```
            End If
        Next
        Estado(3) = "Comp"
        ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & " | "
& Estado(3)
    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Archivo_Valido = False
        Cargando = False
        GetSerialPortNames()                                'Llama al sub procedimiento
GetSerialPortNames
        SerialPort.Encoding = System.Text.Encoding.Default  'Codificación por defecto del sistema
8 bits
        Estado(0) = "Des"
        Estado(1) = "Ning"
        Estado(2) = "Esp"
        Estado(3) = "Esp"
        ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & " | "
& Estado(3)
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        GetSerialPortNames()
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        If Cargando = False Then
            OpenFileDialog.Filter = "Archivos Hex|*.Hex"
            OpenFileDialog.Title = "Abrir archivo Hex"
            OpenFileDialog.FileName = ""
            If OpenFileDialog.ShowDialog() = DialogResult.OK Then
                Ruta = OpenFileDialog.FileName
                'LabelArchivo.Text = OpenFileDialog.SafeFileName
                Estado(1) = "Arch"
                Estado(2) = "Esp"
                Estado(3) = "Esp"
                ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2)
& " | " & Estado(3)
                Archivo()
            Else
                If Ruta = "" Then
                    Estado(1) = "Ning"
                Else
                    Estado(1) = "Arch"
                End If
                Estado(2) = "Esp"
                Estado(3) = "Esp"
                ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2)
& " | " & Estado(3)
            End If
            If Ruta <> "" Then
                Archivo_Valido = True
            End If
        End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Cargando = True
        If Archivo_Valido = True Then
            Estado(2) = "Esp"
            Estado(3) = "Esp"
            ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & "
| " & Estado(3)
            Button3.Enabled = False        'Deshabilita la busqueda de puertos
            ComboBox1.Enabled = False      'Deshabilita la seleccion de puerto
            SerialPort.PortName = ComboBox1.SelectedItem   'El puerto seleccionado del combobox
será al que se conectará
            SerialPort.Open()                   'Abre el puerto seleccionado
```

```
            SerialPort.RtsEnable = True         'Activa el reset del microcontrolador
            Delay_ms(100)                       'Espera 100 ms aproximadamente
            SerialPort.DtrEnable = True         'Habilita la memoria para poder cargar el archivo
Hex
            Delay_ms(100)                       'Espera 100 ms aproximadamente
            SerialPort.RtsEnable = False        'Quita el reset e inicia con la memoria lista
para cargar datos
            Delay_ms(200)                       'Espera 200 ms aproximadamente
            Conexion()                          'Llama al sub procedimiento de conexion
            SerialPort.DiscardInBuffer()        'Descarta todo lo que pueda existir en el bufer
de entrada
            BuferRX = ""                        'Pone BuferRX en vacio
            Verificar = ""                      'Pone Verificar en vacio
            SerialPort.RtsEnable = True         'Activa el reset del microcontrolador
            Delay_ms(100)                       'Espera 100 ms aproximadamente
            SerialPort.DtrEnable = False        'Pone la memoria lista para trabajar como fuente
de código de programa
            Delay_ms(100)                       'Espera 100 ms aproximadamente
            SerialPort.RtsEnable = False        'Quita el reset e inicia con la memoria lista
para trabajar con el programa cargado
            SerialPort.Close()                  'Cierra el puerto de comunicaciones
            Button3.Enabled = True         'Habilita la busqueda de puertos
            ComboBox1.Enabled = True       'Habilita la seleccion de puerto
            ToolStripProgressBar.Value = 0
            Estado(0) = "Desc"
            ToolStripStatusLabelInfo.Text = Estado(0) & " | " & Estado(1) & " | " & Estado(2) & "
| " & Estado(3)
        Else
            MsgBox("No existe un archivo abierto.", MsgBoxStyle.Critical + vbOKOnly, "Error de
archivo")
        End If
        Cargando = False
    End Sub
End Class
```

CÓDIGO FUENTE DEL BOOTLOADER DEL MICROCONTROLADOR

```
#include <mcs51reg.h>
#include<delay_ms.c>
#include<serial-24MHz.c>

__xdata unsigned char * __data p;

static unsigned char conectado;     //0 = no conectado, 1 = conectado
unsigned int direccion;             //direccion de la memoria externa
unsigned char buffer_dato_rx[35];   //buffer para la verificación
unsigned char buffer_dato_tx[35];   //buffer para la grabación en RAM
unsigned char dato_rx;              //caracter recibido por el UART
unsigned char n;                    //puntero a los vectores de buffer
unsigned char n_max;                //cantidad máxima de datos recibidos

void grabar(unsigned int dir, unsigned char dat)
{
   p[dir] = dat;
}
unsigned char leer(unsigned int dir)
{
   return p[dir];
}
void mensaje(unsigned char respuesta)
{
   if(respuesta == 0)
   {
      Serial_Write_Text("CONECTADO\r\n");
   }
   else if(respuesta == 1)
```

```
   {
      Serial_Write_Text("OK\r\n");
   }
}
void conexion()
{
   unsigned char c;

   while(conectado == 0)
   {
      c = Serial_Read_Char();
      if( c == 'c' || c == 'C')
      {
         mensaje(0);
         conectado = 1;
      }
      else if( c == 'v' || c == 'V')
      {
         mensaje(0);
         conectado = 2;
      }
   }
}
void cargar()
{
   while(dato_rx == 0)
   {
      buffer_dato_tx[n] = Serial_Read_Char();
      n_max = buffer_dato_tx[0] - 1;
      n += 1;
      if(n > n_max + 3)
      {
         dato_rx = 1;
      }
   }
   direccion = ((buffer_dato_tx[1] * 256) + (buffer_dato_tx[2]));
   for(unsigned char i = 0 ; i <= n_max ; i += 1)
   {
      grabar(direccion + i,buffer_dato_tx[i + 3]);
      delay_ms(1);
   }
   for(unsigned char j = 0 ; j <= n_max ; j += 1)
   {
      buffer_dato_rx[j] = leer(direccion + j);
      if(conectado == 2)
      {
         Serial_Write_Char(buffer_dato_rx[j]);
      }
   }
   dato_rx = 0;
   n = 0;
   if(conectado == 1)
   {
      mensaje(1);
   }
}
void main(void)
{
   Serial_Init(9600);
   conectado = 0;
   dato_rx = 0;
   conexion();
   while(1)
   {
      cargar();
   }
}
```